

Rechnerunterstützte Analyse und Synthese von Regelungsproblemen mit Mikrorechnern

DISSERTATION

zur

Erlangung des Grades eines

Doktor-Ingenieurs

der

Abteilung für Elektrotechnik

an der Ruhr-Universität Bochum

von

Friedrich Haase

Rheinhausen

Bochum 1985



Digitized by the Internet Archive
in 2026

Rechnerunterstützte Analyse und Synthese von Regelungsproblemen mit Mikrorechnern

DISSERTATION

zur

Erlangung des Grades eines

Doktor-Ingenieurs

der

Abteilung für Elektrotechnik

an der Ruhr-Universität Bochum

von

Friedrich Haase

Rheinhausen

Bochum 1985

Vorwort

Die vorliegende Arbeit entstand während meiner Zeit als wissenschaftlicher Assistent am Lehrstuhl für Elektrische Steuerung und Regelung der Ruhr-Universität Bochum.

Herrn Prof. Dr.-Ing. H. Unbehauen danke ich sehr herzlich für die wertvollen Anregungen, die zahlreichen Diskussionen und die großzügige Förderung während der Durchführung der Arbeit.

Herrn Prof. Dr.-Ing. W. Weber gilt mein besonderer Dank für das der Arbeit entgegengebrachte Interesse und die Übernahme des Mitberichts.

Mein Dank gilt auch allen meinen Kollegen, die mir Gelegenheit gaben zu zahlreichen fruchtbaren Diskussionen und vielerlei Anregungen. Nicht zuletzt gilt mein besonderer Dank Frä. H. Vollbrecht und Frau M.-L. Schmücker für die sorgfältige Anfertigung der Bilder.

Dissertation eingereicht am: 10. April 1985

Tag der mündlichen Prüfung : 15. Mai 1985

Referent : Prof. Dr.-Ing. H. Unbehauen

Korreferent: Prof. Dr.-Ing. W. Weber

Inhaltsverzeichnis

	Seite
1. Einführung	1
2. Mikrorechner und ihre Betriebssysteme .	5
2. 1. Standardbetriebssysteme	8
2. 2. Echtzeitverarbeitung	10
2. 3. Rechnersprachen	14
2. 4. Mikrorechnerarchitekturen	17
2. 5. Zukünftige Entwicklung	18
2. 6. Computergraphik	19
3. Struktur regelungstechnischer Software .	21
3. 1. Aufbau von Programmen	23
3. 2. Programmschnittstellen	25
3. 2. 1. Speicherverwaltung	27
3. 2. 2. Intertaskkommunikation	28
3. 3. Dateienmanagement	30
3. 4. Prozeßankopplung	31
4. Dienstprogramme für regelungstechnische Aufgaben	33
4. 1. Handhabung von Matrizen und Systemen im Zustandsraum	35
4. 2. Handhabung von Übertragungsfunktionen	37
4. 3. Meßwerterfassung	38
5. Identifikation	41
5. 1. Korrelationsanalyse	41
5. 2. Parameteridentifikation	44

6.	Rechnerunterstützter Reglerentwurf . . .	54
6. 1.	Dynamische Reglerentwürfe	54
6. 2.	Regelung mit Zustandsraummethoden . .	62
6. 2. 1.	Zustandsregler	73
6. 2. 2.	Beobachterentwurf	79
7.	Direkte digitale Regelung	90
7. 1.	Implementierung von Reglern	92
7. 2.	Blockorientierte Regelung	95
7. 3.	Zustandsregler	103
7. 4.	Regelung mit Differenzengleichungen .	106
8.	Simulation von Regelkreisen	110
8. 1.	Simulation im Zustandsraum	111
8. 2.	Simulation mit Differenzengleichungen	113
8. 3.	Blockorientierte Simulation	115
8. 4.	Simulation in Echtzeit	123
9.	Zusammenfassung	125

Literaturverzeichnis

Zusammenstellung der wichtigsten Formelzeichen

1. Einführung

Das in dieser Arbeit vorgestellte Softwaresystem soll für eine weit verbreitete Gruppe kleiner Rechner dem Ingenieur zur Lösung regelungstechnischer Aufgaben ein leistungsfähiges und komfortables Werkzeug an die Hand geben. Wichtige Randbedingungen sind Portabilität auf verschiedene Rechnersysteme, leichte Erweiterbarkeit durch den Benutzer und modernes Softwaredesign, um einerseits den Anwender bestmöglich zu unterstützen und um andererseits numerisch zufriedenstellende Methoden zugänglich zu machen. So entstand im Rahmen der vorliegenden Arbeit ein offenes Programmsystem, das leicht den jeweiligen Anforderungen angepaßt oder ergänzt werden kann.

Der Stand der Technik, d. h. die augenblickliche Situation im Bereich der regelungstechnischen Unterstützungssoftware, stellt sich vorläufig noch recht bescheiden dar. Dies zeigt eine kritische Analyse der Beiträge eines Workshop am Institute of Measurement and Control in Brighton über Computer Aided Control System Design im September 1984 [16]. Es wurden dort 29 Beiträge vorgestellt, die über etwa 35 Programme oder Programmsysteme berichten. Hiervon können gerade 6 als Systeme angesehen werden, die einen weiten Bereich von Anwendungen abdecken. Sie erfordern ausnahmslos größere Rechenanlagen von IBM, Digital Equipment, Control Data, Hewlett Packard usw. Im Bereich der etwa 11 mittelgroßen Programmsysteme dominieren bereits spezialisierte Anwendungen. Als Zielrechner wurden hier ebenfalls verhältnismäßig große Geräte der schon erwähnten Hersteller genannt. Nicht ein einziges Beispiel aus diesem Bereich läuft auf einem typischen Personal Computer z.B. einem Apple, IBM-PC oder auf einem Standard CP/M-System. Erst im Bereich der kleinsten und zugleich auch sehr spezialisierten Anwendungen trifft man auf Rechner dieser Gruppe. Etwa 17 Softwarebeschreibungen beziehen sich auf einfache

oder spezialisierte Programme, von denen zumindest 9 bis 12 auf Personal Computern ausgeführt werden können. Größere Programmsysteme, welche einen breiteren Bereich der Regelungstechnik abdecken und dennoch auf einem kleinen Mikrorechner zur Verfügung stehen, existieren derzeit nicht. Gerade die weite Verbreitung kleiner Rechner legt es nahe, sie vielerorts zur Lösung regelungstechnischer Aufgaben heranzuziehen. Sie sind hierzu sehr wohl geeignet, wie in dieser Arbeit gezeigt wird. Das eigentliche Problem scheint eher darin zu bestehen, daß die Mehrzahl anspruchsvoller Programme nicht auf die Möglichkeit einer späteren, breiteren Nutzung hin aufgebaut wurde, sondern nur dem nahe-
liegenden Tagesziel dient. Ein gut ausgebildetes regelungstechnisches Softwaresystem für Mikrorechner wird zur Zeit nirgendwo angeboten. Daß hierfür aber eine sehr große Notwendigkeit besteht, wird allenthalben erwähnt - auch auf dem bereits genannten Workshop [R1].

Die sich so darstellende Situation ist aber keineswegs neu. Man vergleiche hierzu die Aufsätze des IFAC Symposiums über "Computer Aided Design of Control Systems" von 1979 in Zürich [C6]. Bemerkt sei allerdings, daß die Zielsetzung dieses Symposiums etwas anders lag, sodaß eine ähnliche Analyse wie die vorangehende nicht so deutlich ausfallen kann. Nur ein Teil der Beiträge kann hierzu unmittelbar herangezogen werden. Immerhin beschreiben 9 bis 10 Aufsätze Softwarepakete für regelungstechnische Problemstellungen von wechselndem Umfang und sehr unterschiedlicher Anwendungsbreite. Als Zielrechner werden Prozeßrechner oder sogar größere Rechanlagen von DEC, NOVA, UNIVAC oder IBM eingesetzt. Selbst ein Aufsatz der kleine Rechner unmittelbar im Titel führt und eine kurze Analyse der aktuellen Situation (von 1979) gibt, beschreibt ein kleineres Softwaresystem, das für einen mittleren Prozeßrechner entwickelt wurde [A8]. Der angeführte kleinere Tischrechner ist nur in BASIC und in einer assemblerähnlichen Spezialsprache HPL

programmierbar, sodaß eine Übertragung der Software wohl nur durch erneute Codierung erfolgen konnte. Welche weiteren Einschränkungen hierbei hingenommen werden mußten, wird nicht erläutert. Man erkennt jedoch ohne Zweifel, daß der Weg der zukünftigen Entwicklung regelungstechnischer CAD-Software nicht zuletzt auch die Möglichkeiten moderner Mikrorechner berücksichtigen muß.

Für einzelne eng umgrenzte Aufgaben findet man teilweise sehr gute Programme. So wird in [S7] ein BASIC-Interpreter dargestellt, der für eine flexible Implementierung von Meß-, Steuer- und Regelalgorithmen ausgelegt wurde. Abgesehen von der sehr wesentlichen Aufgabe nebenläufige Programme zu ermöglichen, wurden für steuerungstechnische Aufgaben eine Reihe von Funktionen in dieses MSR-Basic aufgenommen. Ein besonderer Vorteil ist die Lauffähigkeit auf einem CP/M-Rechner. Die Arbeit stellt aber eine Sprachentwicklung dar, die sich ausschließlich mit der Aufgabe der Reglerimplementierung beschäftigt. Reglerentwürfe, Simulationen oder die Übernahme von Reglern aus anderen Programmen oder von Dateien sind hierin nicht enthalten. Ein anderes Beispiel ist das blockorientierte Simulationsprogramm PSI [B1, B12]. In geringem Umfang erlaubt es zum Austausch von Daten mit anderen Programmen und kann sogar selbst durch eine Optimierung Regler berechnen. Identifikationsaufgaben oder Entwürfe von Zustandsreglern und Beobachtern werden nicht unterstützt. Als Zielrechner dient eine DEC-Maschine. Auf einem Aussprachetag der VDI/VDE-Gesellschaft Meß- und Regelungstechnik (Langen 1983) findet man eine Reihe weiterer Programmsysteme für mittlere Prozeßrechner und größere Rechenanlagen [G7, I8 u.a.]. Zumindest teilweise versucht man solche Systeme auf Mikrorechnern zu implementieren. Ein anderes Beispiel für die Behandlung von Systemen im Zustandsraum wird in [M7] erwähnt, daß in Versionen für einen Hewlett-Packard Tischrechner und einen IBM-PC vorliegt [J5]. Wenn, wie in diesem Fall, der

Programmumfang mit 500 kBytes nicht zu groß ist [M7], kann man eine Übertragung durchaus denken. Erwähnt sei aber, daß ein Array-Processor an den Mikrorechner zusätzlich angeschlossen werden sollte [J5]. In beiden Versionen wurde als Programmiersprache ein erweitertes BASIC verwendet. Ob und in welchem Umfang Änderungen an den Programmen erforderlich waren, da sich die BASIC-Versionen möglicherweise erheblich voneinander unterschieden, wurde nicht erläutert. Grundsätzlich kann man erkennen, daß zur Zeit vor allem bestehende Programmsysteme von Prozeßrechnern oder gar Großrechnern auf Mikrorechner zu übertragen oder aber kleinere Einzelprogramme für bestimmte Aufgaben neu konzipiert werden.

Das wesentliche Ziel dieser Arbeit ist der exemplarische Aufbau eines CAD-Systems für die rechnergestützte Analyse und Synthese von Regelungsproblemen mit Mikrorechnern. Die Anwendungsbreite des Systems soll hierbei den Entwurf von Regelungen, die Identifikation, die Regelung in Echtzeit und die Simulation von Regelsystemen einschließen. Als Zielrechner ist dabei an kleine Mikrorechnersysteme gedacht, wie sie heute in großer Zahl verwendet werden und voraussichtlich schon in naher Zukunft als "Hand-Held Computer" zur Verfügung stehen werden.

2. Mikrorechner und ihre Betriebssysteme

In einem historischen Rückblick findet man in den fünfziger und sechziger Jahren in erster Linie spezialisierte Betriebssysteme, die für die jeweilige Anwendung und nicht einmal für einen bestimmten Rechner konzipiert und erstellt wurden. Auch heute noch werden solche Betriebssysteme für Sonderaufgaben entwickelt. Für eine breitere Schicht von Anwendungen haben sich aber seither einige Standardbetriebssysteme verbreitet. Für die hier behandelten Mikrorechner gilt dies sogar in größerem Umfang als bei Mini- oder Großrechnern. Lediglich das Betriebssystem UNIX [G6, T1] hat sich in der Gruppe der mittleren und großen Rechner einen festen Platz erobern können.

Für regelungstechnische Aufgaben kann man sich durchaus ein spezialisiertes Betriebssystem vorstellen. Ein entsprechender Versuch für einen Motorola 6800 Mikrorechner [H5] führte zu einem recht leistungsstarken Echtzeitbetriebssystem mit der Möglichkeit, eine ganze Palette von Programmen quasi gleichzeitig ablaufen zu lassen. Durch virtuelle Stackprozessoren für arithmetische Aufgaben und durch Interpreter zur Datenkonvertierung und Formatierung kann man hier einen recht hohen Komfort für eine Programmierung in Assembler erzielen. In jedem Fall aber werden die erstellten Anwenderprogramme sehr spezialisiert sein. An eine Übertragung auf einen anderen Rechner darf man dann natürlich nicht mehr denken. Erst durch den Einsatz von höheren Programmiersprachen gelangt man zu Software, die zumindest in einem gewissem Maße portabel ist - portabel im Sinne von " muß nicht mehr komplett neu geschrieben werden ". Die Verfügbarkeit der Übersetzer für solche Programmiersprachen für Mikrorechner und die zugehörigen Betriebssysteme sind dann ganz wesentliche Voraussetzungen. Von ebenso großer Bedeutung ist es, eine Reihe von Hilfsprogrammen wie Editore, Debugger, Formattierer usw. zu besitzen. Und alle diese

Programme müssen außerdem noch für ein einziges Betriebssystem zur Verfügung stehen, da sie untereinander fast nie austauschbar sind. Erst in den letzten 5 bis maximal 8 Jahren sind diese Voraussetzungen geschaffen worden - wenngleich hier noch viele Wünsche unbefriedigt blieben.

Haben zu Anfang der Mikrorechnerentwicklung noch Assemblerprogramme dominiert, so ist seit einiger Zeit ein steter Anstieg zur Verwendung höherer Programmiersprachen festzustellen. Auch in der Industrie wird in zunehmendem Maße zur Automatisierung und Prozeßsteuerung die eine oder andere Hochsprache eingesetzt. Die Wahl der jeweiligen Sprache hängt dabei oft von der aktuellen Verfügbarkeit und den Kenntnissen der Programmierer ab. Es kann jedoch festgestellt werden, daß man einen deutlichen Trend zu Pascal erkennen kann. Dem wird von anderen Programmiersprachen teilweise dadurch Rechnung getragen, daß man versucht Eigenschaften der Sprachstruktur zu adaptieren. So enthält FORTRAN 77 "IF THEN ELSE" Anweisungen, und es entstanden eine Reihe von "structured" BASIC Versionen.

Auch die Hardware- und Software-Umgebungen werden in zunehmendem Maße standardisiert. Der große Erfolg einer Vielzahl von Programmen und Programmpaketen für administrative Aufgaben ist nicht zuletzt auf identische oder zumindest nahezu gleichartige Hardwarevoraussetzungen und vereinheitlichte Betriebssysteme zurückzuführen. Gerade die Tatsache, daß die Ansprache von Massenspeichern, Sichtgeräten, Druckern usw. in den verschiedenen Betriebssystemen immer über vergleichbare Schnittstellen geleitet wird und mit gleichartiger Funktionsweise verknüpft ist, hat diese Entwicklung unterstützt. Hierbei bleibt im Einzelfall durchaus noch Raum für Variationen, die ein Mikrorechnersystem entweder preiswert (aber meist weniger leistungsstark) oder leistungsfähig (aber meist teuer) machen. So kann als Massenspeicher einmal eine kleine Floppy Disk mit nur 100 kByte, ein anderes Mal eine

Harddisk mit 40 MBytes vorhanden sein, obgleich die Schnittstelle zum Massenspeicher, die von allen Programmen angesprochen werden muß, identisch ist. Ein anderes Beispiel kann die Datenübertragung zu Druckern sein, welche einmal zeichenweise unmittelbar vom Programm selbst erledigt werden muß, oder schneller über einen Puffer und DMA ablaufen kann. Auch die Sichtgerätesteuerung kann unter Verzicht auf eine serielle Übertragung schneller erfolgen. Dies gilt vor allem, wenn der Bildschirminhalt nicht nur innerhalb des Speicherbereiches des Rechners liegt, sondern außerdem unabhängig von der Zentraleinheit einen eigenständigen Controller besitzt, der den Bildaufbau erledigt. Die leistungsfähigsten Beispiele hierfür besitzen eigene Mikroprozessoren einzig für diesen Zweck. Eine solche Auflistung leistungssteigernder Maßnahmen, die keineswegs mit dem zugrundeliegenden allgemeinen Eigenschaften kollidieren, ließe sich nahezu beliebig fortsetzen.

Ein weiterer Punkt für den Erfolg vorgefertigter Software ist in guter Benutzerführung zu sehen. Soll ein Programm durch jemand anderen als seinen Ersteller auf Dauer erfolgreich verwendet werden können, so sind eine Vielzahl von Plausibilitätstests durchzuführen. Derartige Analysen der Eingabedaten können im Einzelfall einen hohen Anteil des Gesamtprogrammes ausmachen und sind oftmals weitaus aufwendiger zu programmieren als der anschließende Algorithmus, der die eigentliche Leistung des Programmes darstellt. Weiterhin müssen vorgefertigte Programme sehr gut dokumentiert sein. Auf die Erstellung akzeptabler Dokumentationen muß sehr viel Wert gelegt werden, woraus ein hoher Anteil des Gesamterstellungsaufwandes rührt. Bei komplexeren Aufgaben eines Programmes sollte man davon ausgehen, daß ein Drittel und mehr der Arbeit für die Dokumentation anfällt. Darüber hinaus kann die Dokumentation auch kaum an andere Personen delegiert werden. Eine verständliche Beschreibung eines aufwendigen Programms erfordert den gleichen Sachverstand wie das Programm

selbst und muß daher gemeinsam mit dem Programm erstellt werden.

Eine Besonderheit stellen zweckorientierte Betriebssysteme dar, wie sie für spezialisierte Aufgaben erstellt werden [U2]. Eine fertige Applikation benötigt meistens kein komplettes Betriebssystem mehr. Es ist eher ein einzelnes Programm, welches in einer entsprechenden Hardwareumgebung ausschließlich noch die Funktionen realisiert, die zum Betrieb erforderlich sind [M2]. Solche Betriebssysteme werden der Aufgabe entsprechend konstruiert und sollen hier nicht weiter erwähnt werden.

2. 1. Standardbetriebssysteme

Mitte der siebziger Jahre stießen Mikrorechner in den Leistungsbereich vor, der notwendig war, um sie in administrativen Bereichen einsetzen zu können. Hierzu waren Betriebssysteme erforderlich, die einfach und preiswert waren [B3]. CP/M von Gary Kildall später Digital Research hat sich zumindest für den Bereich der 8 Bit Rechner als das Standardbetriebssystem herausgestellt. Für dieses System gibt es weltweit die meiste Software für praktisch jeden Anwendungsbereich, sofern er von allgemeinerem Interesse ist, sodaß ihre Entwicklungskosten auf eine größere Zahl von Anwendern verteilt werden konnte. Im Bereich der 16 Bit Mikrorechner sind zur Zeit zwei Betriebssysteme sehr weit verbreitet. Eines ist die entsprechend geänderte Version von CP/M unter dem gleichen Namen. Ein zweites ist MSDOS von Microsoft, bzw. PCDOS wie es in der Version für den IBM-PC genannt wird. Beide sind in mancherlei Hinsicht sehr ähnlich, aber leider nicht miteinander verträglich. Ob in Zukunft beide nebeneinander bestehen werden, oder ob eines von beiden

verschwinden wird, bleibt abzuwarten. Zumal IBM für die Nachfolgemodelle ihres PCs neben PCDOS (MSDOS) auch XENIX - eine UNIX-Version - anbietet. Dieses Betriebssystem stammt wiederum von Microsoft, die von Bell eine UNIX-Lizenz besitzt. Der bisherige Erfolg von IBM verleitet zu der Vermutung, daß MSDOS und XENIX für die nähere Zukunft zum Standard werden.

Steigende Leistungsfähigkeit auch bei den Mikrorechnern führte bis heute zu einer ständigen Assimilation von UNIX-typischen Eigenschaften auch bei Mikrorechnerbetriebssystemen. So gibt es heute sogar eine ganze Reihe von UNIX look-alike Systemen wie CROMIX [C2] oder auch Xenix.

Für eine etwas breitere Aufgabenstellung im Bereich der Regelungstechnik wird man auf eine typische Konfiguration mit zwei Programmen stoßen. Eines dieser Programme erledigt die eigentliche Regelaufgabe. Es muß zyklisch entsprechend der Abtastzeit gestartet werden, führt mehr oder weniger umfangreiche Berechnungen aus, greift hierzu auf die Anschlüsse zur Prozeßschnittstelle zu und wartet nach Erledigung seiner Aufgabe passiv bis zum nächstfolgenden Abtastzeitpunkt. Es wird also insbesondere keinerlei Ein-/Ausgabeoperationen über ein angeschlossenes Sichtgerät abwickeln. Bis ein Bediener des Programms etwa einen neuen Sollwert eingegeben hat, ist der Prozeß längst weggelaufen. Eine schritthaltende Bearbeitung der Regelung ist nur möglich, wenn alle langsamen Vorgänge in ein Steuerprogramm verlagert werden. Dieses Steuerprogramm kann nun in interaktiver Form die Kommunikation mit dem Benutzer übernehmen. Eingetretene Veränderungen von Sollwerten usw. werden nach Bedarf an das Regelprogramm weitergeleitet.

Daraus folgt unmittelbar, daß ein Betriebssystem mit einem Multitasking notwendig ist. Für 8 Bit- und 16 Bit-Mikrorechner stehen hierfür seit längerem MP/M [D1] und erst seit kürzerer Zeit Concurrent CP/M

[G3] zur Verfügung. Beide sind erweiterte CP/M Versionen. MP/M ist dabei in der Lage, neben mehreren Programmen auch noch mehrere Konsolen zu verwalten - es erlaubt also ein Multiprogramming. Im Rahmen dieser Arbeit wird hiervon aber kein Gebrauch gemacht. Beide Betriebssysteme ermöglichen außerdem Speichererweiterungen über ein Banking des Hauptspeichers.

Zu Beginn dieser Arbeit lag nur eine 8 Bit MP/M-Version als Betriebssystem vor, sodaß diese zwangsläufig verwendet werden mußte. Da aber Concurrent CP/M keine wesentlich neuen und auch keine dringend erforderlichen zusätzlichen Eigenschaften bot, wurde es bisher nicht eingesetzt. Die Übertragung eines Teiles der vorhandenen Software auf ein 16 Bit System unter MSDOS zeigt, daß die Portabilität weitgehend sichergestellt ist. An den übertragenen Programmen waren praktisch keinerlei Änderungen durchzuführen.

2. 2. Echtzeitverarbeitung

Auf die Notwendigkeit von Parallelarbeit durch zwei Programme zur Behandlung regelungstechnischer Aufgaben wurde bereits hingewiesen. Eines dieser beiden Programme muß der Abtastzeit entsprechend zyklisch gestartet werden und terminiert sich jeweils anschließend selbst, um bis zum nächsten Abtastzeitpunkt zu warten. Hierzu ist eine Zeitverwaltung notwendig. Echtzeitbetriebssysteme besitzen oftmals eigens hierfür bereitgestellte Grundfunktionen. In den üblichen Betriebssystemen für Mikrorechner sind diese nicht unmittelbar vorgesehen.

Um Echtzeitprogramme zu implementieren, wird eine Verwaltung benötigt. Die zwingend notwendigen Funktionen dieser Echtzeitverwaltung besteht aus dem

zyklischen Start des Echtzeitprogramms, aus dem endgültigen Beenden des Programms, wenn dies erforderlich ist, und einem beliebig lang andauernden Anhalten des Programms, wenn das Echtzeitprogramm zwar vorhanden, aber noch nicht oder nicht mehr betriebsbereit ist [H1]. Um zu erkennen, ob die gewählte Abtastzeit wirklich ausreichend ist, muß außerdem noch ein Merker behandelt werden, der gesetzt wird, sobald die Rechenzeit des Echtzeitprogramms die Abtastzeit übersteigt. In diesen Fällen liegt immer ein Fehlerzustand vor. Er kann im einfachsten Fall durch den Entwurf einer Regelung mit einer höheren Abtastzeit gelöst werden. Ist dieses nicht möglich, so muß man versuchen die Rechenzeit zu verringern. Denkbar ist hier eine zweckmäßigere Formulierung des Algorithmus oder auch ein Übergang zu einem anderen Datenformat, welches bei geringerer Genauigkeit eine höhere Verarbeitungsgeschwindigkeit ermöglicht. Für allgemeine Regelprogramme ist dies aber selten ein gangbarer Weg. Die Normierung eines Algorithmus, die erforderlich ist, um mit einer Festpunktdarstellung von Zahlen zu arbeiten, ist nur für einen jeden Anwendungsfall gesondert durchzuführen. Die Detektierung des erwähnten Fehlers aber muß auf jeden Fall gewährleistet sein, da er sonst unentdeckt bleiben könnte.

Bei dem Betriebssystem MP/M konnte ein Echtzeitanschluß über die dort vorhandene Echtzeituhr verhältnismäßig leicht implementiert werden. Wäre dies nicht der Fall, so müßte eigens zu dieser Verwaltung ein Zeitgeberbaustein angeschlossen werden. Das Echtzeitverwaltungsprogramm hätte diesen zu initialisieren und eine Interruptserviceroutine zu installieren. Im einfachsten Fall könnte unmittelbar von dieser Routine aus ein Aufruf des Reglerprogramms erfolgen. Sofern das Betriebssystem dies zuläßt, erscheint es aber besser, eine normale Programmumschaltung durchzuführen. Im System MP/M geschieht dies über Flags, die vom Betriebssystem verwaltet werden, auf die Programme warten können und die durch

Interruptserviceroutinen gesetzt werden können. Hiermit kann in MP/M auch gleich eine Prioritätsverteilung für die Programme eingestellt werden.

Neben dieser Art von Unterbrechungen kann es in einem System weitere Interrupts geben, die von den Anwenderprogrammen selbsttätig verwaltet werden müssen. Ein typisches Beispiel liegt in der Ermittlung des Zündzeitpunktes der Thyristoren eines Gleichstrommotors vor [M3]. Hier kann mit zwei Unterbrechungen der Ablauf eines Programms zum Öffnen der Thyristoren angestoßen werden. Ein erster Interrupt erfolgt immer im Spannungsnulldurchgang der jeweiligen Netzphase. Im angeschlossenen Interruptservice wird dem erforderlichen Zündwinkel entsprechend ein Zeitgeber gestartet, welcher nach Ablauf der eingestellten Zeit und damit auch zum Zeitpunkt einer entsprechenden Phasenlage der Netzspannung einen weiteren Interrupt erzeugt. Als Interruptservice dieses zweiten Interrupts können dann die Zündimpulse für den Stromrichter erzeugt werden [J4].

Abgesehen von der Lösung spezieller Probleme eines angeschlossenen Prozesses unter Verwendung solcher Unterbrechungen, kann man für eine schritthaltende Bearbeitung typische Eigenschaften für eine Prozeßregelung fordern. Diese sind zwar für Mikrorechner nicht spezifisch, sondern gelten ebenso für jeden größeren Prozeßrechner, doch kann der Einsatz der Mikrorechnertechnologie zu teilweise andersartigen Lösungen führen, als sie bei Prozeßrechnern heute realisiert sind. Eine einfache und oft ausreichend leistungsfähige Programmarchitektur erfordert ein Echtzeitprogramm, das der Abtastzeit entsprechend den eingesetzten Regelalgorithmus beinhaltet, und ein Steuerprogramm, welches diesen Algorithmus mit den erforderlichen Parametern versorgt. In einer Anwendung können zu diesen beiden Tasks zur Prozeßdatenerfassung, Grenzwertüberwachung, An- und Abfahren des Prozeß usw. weitere hinzukommen.

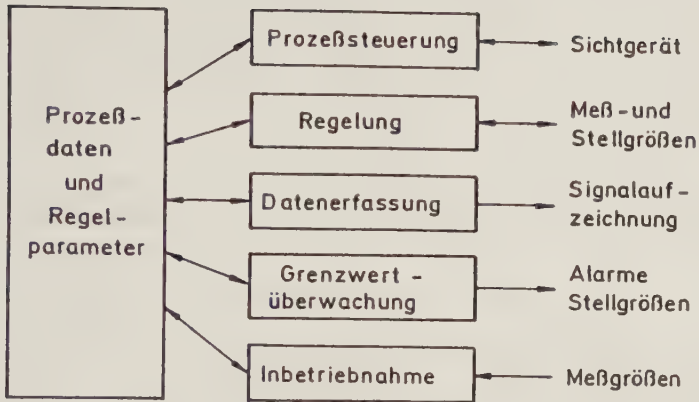


Bild 2.2.1 Struktur einer Prozeßautomatisierung

In Bild 2.2.1 sind die angeführten Teilaufgaben zusammen gezeigt. Jedes Teilprogramm stellt seiner jeweiligen Aufgabe entsprechend eine Verbindung zwischen den angeschlossenen Prozeßteilen - Sichtgerät, Meß- und Stellgrößen - und einem gemeinsamen Datenbestand dar. Zur Realisierung dieser Struktur stehen prinzipiell zwei Wege offen. Man bedient sich entweder einer einzigen Zentraleinheit und eines Multitasking-Betriebssystems, oder die Prozeßautomatisierung wird auf mehrere Prozessoren aufgeteilt, die auf gemeinsame Daten zugreifen dürfen. Die Kopplung der Teilprogramme über die gemeinsamen Daten kann ebenfalls verschieden ausgeführt sein. Verwendet man allen Prozessoren gleichermaßen zugängliche Speicherbereiche, so liegt eine feste Kopplung vor. Kommunizieren die Programme durch Übertragen der Daten, so liegt eine lose Kopplung vor.

Von Standardbetriebssystemen wird zur Zeit überwiegend die erstgenannte Methode, d.h. Multitasking mit nur einer Zentraleinheit unterstützt. Es ist allerdings eine zunehmende Verbreitung von Systemen mit verteilter Rechenleistung zu erkennen. Ein gut ausgebildetes Softwaresystem kann auch solche Problemlösungen durch

die Bereitstellung vorgefertigter Teile unterstützen, die in die einzelnen Programme integriert werden können.

2. 3. Rechnersprachen

Für die heute meistverbreiteten Standardbetriebsysteme sind praktisch alle wesentlichen Programmiersprachen verfügbar. Oftmals hat man sogar unter mehreren Implementierungen eine Auswahl. Damit stellt sich die Frage nach einer geeigneten Sprache zur Realisierung eines größeren Programmsystems für regelungstechnische Aufgaben. Zwar wäre hier prinzipiell fast jede Rechnersprache geeignet, doch kann eine schlechte Wahl zu erheblichem Mehraufwand in der Programmerstellung und Pflege führen. So steht außer Frage, daß mit Assembler alle Wünsche an ein Programm bestmöglich erfüllbar sind. Es steht aber ebenfalls außer Frage, daß Assemblerprogramme extrem zeitaufwendig in der Programmierung und Änderung sind, und daß sie nur durch eine fast vollständige Neukodierung von einem Rechner auf einen anderen Übertragen werden können. Von dieser Möglichkeit wurde daher Abstand genommen. Assemblerprogramme wurden nur in ganz geringem Umfang eingesetzt und auch nur dort, wo ein unmittelbarer Zugang zur Rechnerhardware unumgänglich ist.

Ein zweiter wesentlicher Punkt bei der Auswahl der Programmiersprache waren die unterschiedlichen Möglichkeiten zur Behandlung von Text. Für eine leistungsfähige interaktive Handhabung komplexer Programme muß schon von der Sprache her die Behandlung leicht zugänglich sein. Diese Forderung spricht für den Einsatz von Pascal [M1] oder - wenngleich in geringerem Maße - für einen modernen BASIC-Dialekt. Die ältere und auch heute noch sehr viel verwendete Sprache FORTRAN ist zur

Behandlung von Text weniger geeignet. Die Handhabung von Daten ist in FORTRAN allerdings umfangreicher. Insbesondere enthält sie im Gegensatz zu Pascal komplexe Variablen und erlaubt die zugehörigen Operationen. Eine grobe Abschätzung des Aufwandes für komplexe Arithmetik einerseits und der Programmteile für interaktive Behandlung andererseits spricht eindringlich für Pascal. Fast alle Programme des vorgestellten Systems sind interaktiv, wohingegen komplexe Arithmetik weitaus seltener zu finden ist.

Ein letzter Punkt, der zur Wahl der Programmiersprache Pascal beigetragen hat, war der Wunsch nach einer strukturierten Programmierung. Sie besitzt den erheblichen Vorteil, daß eine Vielzahl von immer wieder auftretenden Programmierfehlern bereits durch den Compiler erkannt wird. Die sehr viel angenehmere Handhabung der strukturierten Programmierung stellt dazu eine wertvolle Ergänzung dar. Eine andere Möglichkeit wäre eine der sogenannten Echtzeitsprachen gewesen. Für die einzusetzenden Standardbetriebsysteme stehen bislang aber nur einige BASIC-Erweiterungen zur Verfügung, die aber nicht ausreichend leistungsfähig erschienen. Echtzeitsprachen wie Pearl [W2] oder Ada sind derzeit nicht oder noch nicht verfügbar, sollten aber in späteren Ergänzungen in Betracht gezogen werden.

Ada ist eine strukturierte Rechnersprache, die sehr viel Ähnlichkeit mit Pascal besitzt. Ihr ganz grundlegender Unterschied zu Pascal besteht darin, mehr als eine einzelne Task formulieren zu können. Als Echtzeitsprache unterstützt Ada ein Multitasking. Die jeweiligen Teilaufgaben werden in Ada in einzelnen Tasks ausgeführt, und erst die Gesamtheit aller zu einem Programm gehörenden Tasks löst die gestellte Aufgabe. Neben diesem Multitasking gibt es noch einige weitere Eigenschaften, die in Ada weit besser gelöst sind als in Pascal. Leider sind Ada-Übersetzer zur Zeit nur für wenige Rechnersysteme verfügbar. Insbesondere

für die hier vorausgesetzten Standardbetriebssysteme wird man noch einige Zeit auf Adacompile warten müssen, da die Sprache selbst - ähnlich wie auch Pearl - ein sehr leistungsfähiges und auch spezialisiertes Betriebssystem voraussetzt. Eine spätere Übertragung von Pascalprogrammen nach Ada ist sehr leicht möglich, wenn man als Vergleich etwa die Schwierigkeiten heranzieht, welche die Übertragung eines FORTRAN-Programms nach Pascal in der Regel macht. Ein erheblicher Teil der Programmänderungen für eine Umsetzung von Pascal nach Ada könnte sogar automatisiert erfolgen, da sich Ada und Pascal oftmals nur in den Schlüsselwörtern unterscheiden [W1]. Vergleichbares gibt es für eine Umsetzung von FORTRAN nach Pascal nicht. Hier wäre eine weitgehende Neucodierung erforderlich. Betreffend der Verwendung gemeinsamer Daten in zumindest zwei Programmen, ist Ada als Entwicklungssprache für Echtzeitprogramme sicherlich ein leistungsfähigeres Werkzeug als Pascal. Die Übertragung von Informationen wie etwa neuen Sollwerten oder geänderten Koeffizienten einer Übertragungsfunktion vom Steuerprogramm zum Echtzeitprogramm ist in Ada sehr einfach zu formulieren und sofort im Kontext verstehbar. Vor allem hier würden sich eine Reihe von Vorteilen ergeben. Nun muß man aber berücksichtigen, daß ein erheblicher Teil der Software eines regelungstechnischen CAD-Systems keine Echtzeitbearbeitung erfordert. In diesem Bereich, der unter anderem fast alle Entwurfsprogramme für Regler enthalten wird, ist mit Ada keine unmittelbare Leistungssteigerung zu erwarten. Vielmehr wäre es wünschenswert, daß alle Programme zumindest miteinander verträglich sind, d.h. daß beliebige Rechnersprachen verwendet werden können. Bisher haben unterschiedliche Datenformate vor allem von REAL-Größen hier Probleme bereitet [A4, A5]. Mit der Einführung von arithmetischen Coprozessoren und einer Standardisierung der Datenformate [IEEE P754] sind hier Verbesserungen in Sicht und teilweise schon verbreitet [I5].

Durch den zunehmenden Einsatz kleiner Rechner in

allen technischen Bereichen kommen ständig neue Programmierwerkzeuge auf den Markt, sodaß man in Zukunft noch Verbesserungen hinzugewinnen kann. Die Entwicklungstendenzen gehen dabei stark in Richtung strukturierter Sprachen, sodaß die Wahl von Pascal auch auf Dauer eine gute Ausgangsbasis bleiben wird.

2. 4. Mikrorechnerarchitekturen

Mikrorechnerarchitekturen sind beim Aufbau eines größeren Softwaresystems nur bedingt von Bedeutung. Bei der Erstellung von Programmen sind sie nicht erkennbar, da von einer höheren Programmiersprache die Zentraleinheit natürlich unzugänglich ist [B2]. Aber bereits durch die Größe eines einzelnen Programms kann die Leistungsfähigkeit einer CPU deutlich zutage treten [F1]. Es wurde streng darauf geachtet, daß alle Programme dieses Systems auch auf einem kleinen Standard CP/M-System arbeitsfähig sind. Da solche Systeme in absehbarer Zeit so klein sein werden, daß sie sogar in einer Aktentasche Platz finden, ist zu erwarten, daß das hier vorgestellte Programmsystem entsprechend eingesetzt werden wird. Einfache Vorläufer zu solchen Geräten sind inzwischen verfügbar.

Ein weiterer Punkt, an dem sich die Überlegenheit der modernsten Prozessoren widerspiegelt, sind die in den Programmen erforderlichen Rechenzeiten. Bei sehr aufwendigen Algorithmen und einer der älteren Zentraleinheiten wie z.B. dem Z80 dauern Reglerentwürfe oftmals recht lange. Meist müssen die Berechnungen einmal durchgeführt werden, sodaß die Rechenzeiten erträglich bleiben. Bei Echtzeitprogrammen hingegen muß die erzielbare Rechenleistung streng beachtet werden. Vor allem bei sehr schnellen Prozessen kann es erforderlich werden, leistungsfähigere Mikrorechner zu

verwenden und Arithmetikprozessoren einzusetzen [F3].

Bis zu einem gewissen Grade muß man hier auch die zur Zeit verfügbaren Compiler kritisieren. Wenn man sich den von ihnen erzeugten Programmcode genauer ansieht, so kann man erkennen, daß sie nicht immer die mögliche Leistung der CPU nutzen. Es ist zu vermuten, daß die Compiler in erster Linie möglichst schnell auf den Markt kommen sollten und deshalb sehr einfach aufgebaut wurden. Die Qualität des erzeugten Programmcodes wie sie bei größeren Rechenanlagen allenthalben gewährleistet ist, bleibt offenbar späteren Compiler-versionen vorbehalten.

2. 5. Zukünftige Entwicklung

Als Weiterentwicklungen der Mikrorechnerbetriebssysteme sind in erster Linie eine Unterstützung des Multitasking, verbessertes Dateienmanagement und die bessere Verwaltung großer Massenspeicher zu sehen. Zur Echtzeitprogrammierung von Regelalgorithmen wäre ein gutes Multitasking sehr hilfreich. Die Möglichkeiten, die schon heute von MP/M oder Concurrent CP/M geboten werden, sind allenfalls gerade ausreichend. Die Verbesserung von Coprozessoren und ihre Zusammenarbeit mit der CPU wird die Leistung von Mikrorechnern weiter steigern [I2, I3, I4], ohne den Umfang und Preis solcher Systeme erheblich zu verändern.

In weitaus stärkerem Umfang wird sich die verfügbare Software für Mikrorechner erneuern. Editoren, Compiler und andere Werkzeuge werden ständig verbessert. Zugleich aber ist festzustellen, daß auch spezialisiertere Programme oder Programmsysteme weite Verbreitung finden. Es ist durchaus vorstellbar und wünschenswert, daß das hier vorgestellte Softwaresystem

für regelungstechnische Problemstellungen zu einem späteren Zeitpunkt den Kern eines Standardprogrammsystems darstellt.

2. 6. Computergraphik

Viele Programme, die regelungstechnische Aufgaben lösen, erfordern eine graphische Darstellung der erhaltenen Resultate. Oftmals ist es sogar der eigentliche Zweck eines Programms, eine solche Darstellung zu berechnen. Ein typisches Beispiel sind hierfür digitale Simulationen, die den zeitlichen Verlauf der Signale einer Regelung berechnen. Von seltenen Ausnahmen abgesehen, ist der Regelungstechniker an den einzelnen Werten selbst weniger interessiert. Erst ihre Gesamtheit in der Form eines Bildes gibt den gewünschten Aufschluß über das Verhalten des Regelsystems.

Es mußten daher graphische Unterstützungsprogramme erstellt werden, die eine möglichst breite Anwendung in den verschiedenen Teilprogrammen finden konnten [A6]. Leider sind solche Programme für Mikrorechner zur Zeit noch nicht mit den Leistungen verfügbar, die man für eine allgemeine Benutzung erwartet. Der steigende Anteil an graphischen Programmen der kommerziellen Software wird diese Situation bald verbessern. Die Verwendung der vorliegenden Graphik konnte so ausgelegt werden, daß der Einsatz leistungsfähiger Graphik, wie beispielsweise dem genormten GKS [E1], zu keinen übermäßigen Änderungen führen sollte.

Von ihrer Struktur her ist die Graphik des entstandenen Programmsystems in der Form von Modulen benutzbar, die nur die jeweils benötigten Daten durch das Anwenderprogramm erhalten. Es ist gegliedert in einen Teil, der vom jeweils verwendeten Ausgabegerät

unabhängig ist und alle erforderlichen Transformationen der verschiedenen Koordinatensysteme durchführt, und einen Teil, der für das Ausgabegerät spezifisch ist. Eine solche Unterteilung ähnelt den Workstations von GKS. Als Ausgabegeräte sind zur Zeit ein nicht-graphisches Sichtgerät mit einer "Sternchengraphik", ein Sichtgerät mit Feingraphik und ein Plotter implementiert. Durch die Gliederung des Graphikprogramms kann jedes andere Ausgabesystem ohne großen Aufwand angesprochen werden. Es ist nur der vom Umfang recht geringe, geräteabhängige Teil zu modifizieren.

3. Struktur regelungstechnischer Software

In Bild 3.0.1 sind die Stationen einer regelungstechnischen Aufgabe dargestellt. Ausgangspunkt ist zumeist eine reale Regelstrecke, für die ein einsatzfähiger Regler entworfen und realisiert werden soll. Die zu diesem Zeitpunkt vorliegende Beschreibung besteht meist aus einem mehr oder weniger korrekten mathematisch/physikalischen Modell. Zur Formulierung des regelungstechnischen Problems ist eine Analyse dieses Systems notwendig. Nicht immer kann man eine Zustandsraumdarstellung oder eine Übertragungsfunktion des Systems unmittelbar angeben, sodaß man erst über Messungen an der Regelstrecke hierher gelangt. Denkbar ist eine "On-Line"-Identifikation mit den natürlichen Signalen des Prozesses oder auch unter Zuführen zusätzlicher Testsignale. Andererseits können solche Signale zunächst nur erfaßt und auf einem Massenspeicher abgelegt werden und anschließend durch eine "Off-Line"-Identifikation ausgewertet werden. Der Vorteil des zweiten Weges besteht in der Möglichkeit, mit exakt dem gleichen Satz von Meßwerten unterschiedliche Identifikationsverfahren einzusetzen. Die erhaltenen Resultate können durch Simulation des Verhaltens der Regelstrecke gegebenenfalls verifiziert werden.

Der nächste Schritt ist die Synthese. Hier wird für das regelungstechnische Problem eine passende Lösung gesucht. Mit Reglerentwürfen im Frequenzbereich, Zustandsreglerentwürfen und Beobachterentwürfen wird man versuchen, die erwünschten Eigenschaften des geschlossenen Regelkreises zu erzielen. Auch hier kann durch eine Simulation des Regelkreises und des Reglers eine Überprüfung erfolgen. Dabei anfallende Signale können auch zu weiteren Untersuchungen eingesetzt werden, etwa indem man die Eigenschaft der Regelung durch eine Korrelation spektral beurteilt.

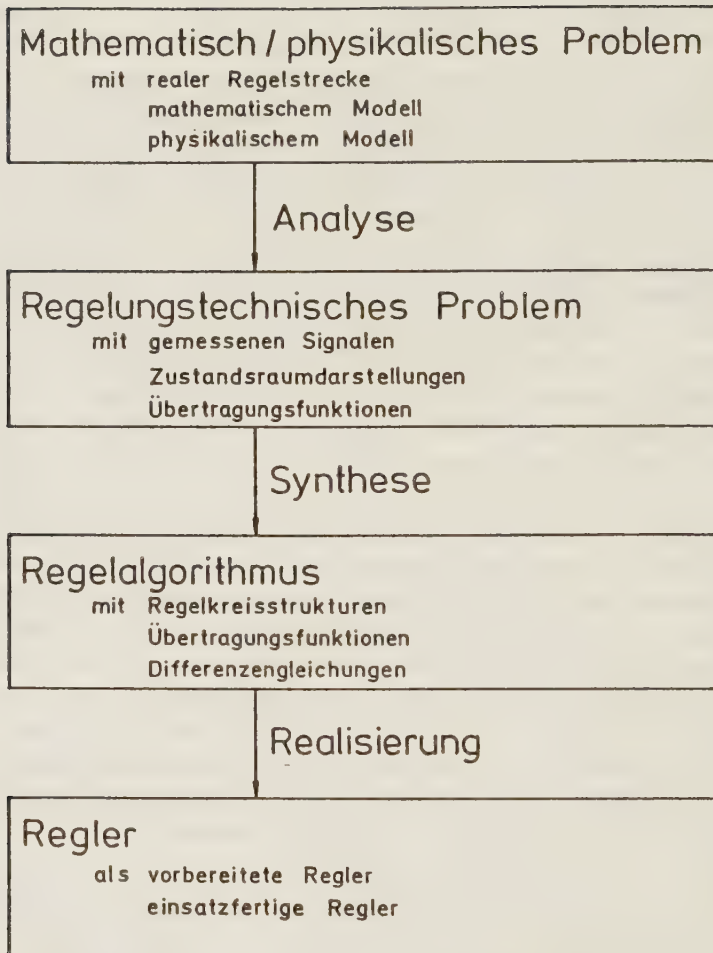


Bild 3.0.1 Stationen einer regelungstechnischen Aufgabe

Als Ergebnis der Regelung fallen Regelkreisstruktur und Regler in der Form von Übertragungsfunktionen oder Rückkopplungsmatrizen an. Falls eine diskrete Regelung angestrebt wurde, handelt es sich dabei letztlich um Differenzengleichungen, die im "On-Line"-Betrieb verarbeitet werden müssen. Zur Realisierung wird man zunächst die vorbereiteten Regler auf

dem gleichen Rechner testen. Interaktive Eingriffe zur Änderung von einzelnen Parametern und die Aufzeichnung von kennzeichnenden Signalen sind an dieser Stelle noch möglich. Erst wenn sich die entworfene Regelung als zufriedenstellend herausgestellt hat, wird man daran gehen, den einsatzfertigen Regler zu konstruieren. Dieser wird dann aber nicht mehr auf dem gleichen Rechner ablaufen, sondern auf einem Applikationssystem, das nur noch jene Funktionen zuläßt, die in der Aufgabenstellung gefordert wurden. Signalwerte innerhalb des Systems oder andere Prozeßzustände und Parameter sind bei einem solchen fertigen Regler nur dann oft nicht mehr zugänglich.

Unberücksichtigt blieb im Bild 3.0.1 allerdings, daß die einzelnen Stationen eines Entwurfs einer Regelung oft mehrmals durchlaufen werden, daß die ingenieurmäßige Aufgabe also mit einer gewissen Anzahl von Versuchen verbunden ist und nicht rezepturartig sofort zu dem gewünschten Resultat führt.

3. 1. Aufbau von Programmen

Das in dieser Arbeit dargestellte System besteht aus einer Sammlung untereinander verzahnter Einzelprogramme für die wichtigsten analytischen und synthetischen Aufgaben der Regelungstechnik. Bei seiner Erstellung wurden eine Reihe von Forderungen zur allgemeinen Grundlage gemacht. Zum einen findet man hier solche Regeln, die dem Anwender des Systems einen geschlossenen Gesamteindruck geben. Gleichgültig an welcher Stelle der Programme er sich befindet, er kann für ein ganzes Spektrum von Einzelaufgaben ähnliche Reaktionen des Rechners erwarten. Dieser Punkt kann als eine ergonomische Gestaltung der Programme betrachtet werden und stellt selbst einen hohen Wert dar, da er

für die Handhabbarkeit der Systemteile und die Akzeptanz von wesentlicher Bedeutung ist. Zum anderen sind die Randbedingungen aller Programme an den heutigen und in naher Zukunft vorzufindenden Möglichkeiten kleiner Rechner orientiert. Die Gliederung des Systems und die verschiedenen Wege durch seine Teilprogramme werden hiervon beeinflußt und bedurften einer genauen Planung.

Ein zweiter wesentlicher Punkt bei der Gestaltung der Programme dieses Systems liegt in der zweckmäßigen Aufteilung der Einzelaufgaben. Anders als bei mittleren und großen Rechenanlagen müssen Programme für Mikrorechner tiefer gegliedert werden, da sie engeren Grenzen der verfügbaren Speicher unterliegen. Die nachfolgend dargestellten vielfältigen Möglichkeiten des Systems zeigen deutlich, daß trotz solcher Beschränkungen sehr leistungsfähige Systeme für regelungstechnische Aufgaben erstellt werden können. Mikrorechner, die heute bereits nahezu am unteren Ende der Größenskala stehen und preislich auch heute schon für jedermann erschwinglich sind, bieten dem Regelungstechniker über ein entsprechendes Programmpaket eine außerordentliche Hilfe bei seiner Arbeit.

Zuletzt sei hier noch angeführt, daß allen Programmen die Beschränkungen auferlegt wurden, die eine leichte Übertragung auf andere Rechner erlaubt. Von ganz wenigen Programmteilen abgesehen kann jede UCSD-ähnliche Version von Pascal zur Implementierung verwendet werden [Z1]. Gegenüber dem zunächst genormten Pascal [J1] wurden Variablen vom Typ STRING eingesetzt, die in der ursprünglichen Formulierung nicht vorhanden waren, die aber heute in nahezu jeder Pascalversion vorhanden sind und für die inzwischen ebenfalls eine ISO-Norm vorliegt.

3. 2. Programmschnittstellen

In nahezu allen Programmen liegt eine Struktur als Ausgangsbasis vor, die nicht allein für das jeweilige Programm spezifisch ist, sondern zugleich eine Basis darstellt, welche für eine Gruppe von Programmen identisch ist. So sind die zum Teil erheblichen Datenmengen, die für Reglerentwürfe oder Simulationen erforderlich sind, klassifiziert und in jeweils passenden Dateien enthalten. Eine Erweiterung des Systems etwa durch einen neuen Regler- oder Beobachterentwurf kann daher verhältnismäßig schnell implementiert werden, da eine Vielzahl von Dienstleistungen an anderer Stelle vorliegen und übernommen werden können. Um die Integrität des Systems als Ganzem aufrechtzuerhalten, ist es sogar zwingend erforderlich, auf solche Moduln zurückzugreifen. Andernfalls wäre die Kompatibilität der Teile des Systems untereinander nicht mehr gegeben. Da diese Schnittstellen zwischen den Programmen meist einfacher und allgemeiner verwendbar sind, bedeutet ihre Verwendung für jeden Anwender natürlich eine erhebliche Arbeitserleichterung.

Allen Programmen gemeinsam ist eine ähnliche Benutzerführung. In allen Programmen werden Meldungen des Systems, die die Fortsetzung der Arbeit oder eventuelle Fehler des Benutzers kenntlich machen, durch Klartextausgaben dargestellt. Die bei kleinen Rechnern oft unangenehm starken Einschränkungen hinsichtlich der Größe des Hauptspeichers und der angeschlossenen Massenspeicher erfordern allerdings manchmal sehr knapp gehaltene Meldungen. Aus diesem Grund wurden zentrale Hilfsmenues vorgesehen, welche die Teilleistungen eines Programms auflisten. Genauere Erläuterungen findet man in jeweils angeschlossenen "On-Line"-Manuals, die auf den angeschlossenen Massenspeichern vorliegen und in ihrem Umfang der jeweils vorhandenen Speicherplatzmenge dort angepaßt werden können. Hiervon abgesehen existiert zu jedem Programm eine ausführliche Dokumen-

tation. Wegen ihres Umfangs dauert ein Zugriff auf Informationen auf diesem letzten Wege natürlich am längsten. Doch darf sie andererseits besonders umfangreich und detailliert sein, da sie keiner der erwähnten Beschränkungen unterliegt. In diesen Dokumentationen sind sogar exemplarische Beispiele zu finden, die den Einsatz der Programme und die zugrundeliegenden Regelstrukturen erklären und diese selbst mit ihren Voraussetzungen und Verfahrensvorschriften kommentieren, wo es für eine Erweiterung der Programme erforderlich erscheint.

Eine weitere wichtige Eigenschaft aller Programme sind ständige Überprüfungen aller vorliegenden Eingabedaten auf Konsistenz. Diese Plausibilitätstests bewirken, daß ein einmal begonnener Weg folgerichtig weiterbeschritten werden muß, sofern man ihn nicht wie im Falle eines Irrtums abbricht und erneut beginnt. Als ein einfaches Beispiel sei die Eingabe einer Übertragungsfunktion erwähnt. Der zunächst definierten Ordnung des Systems entsprechend wird das Programm eine eindeutige Anzahl von Koeffizienten oder Pol- und Nullstellen erwarten. Ein anderes Beispiel ist ein Reglerentwurfsprogramm, welches eine diskrete Regelstreckenübertragungsfunktion voraussetzt. Es wird niemals die Koeffizienten einer auf einer Datei abgelegten kontinuierlichen Übertragungsfunktion akzeptieren. Eigenschaften dieser Art entsprechen dem heutigen Stand der Softwaretechnik und müssen in jedem Programm anzutreffen sein, welches von anderen als dem ursprünglichen Ersteller verwendet werden soll.

3. 2. 1. Speicherverwaltung

In der Grundversion erfordern die meisten hier vorgestellten Programme einen CP/M-Rechner. Lediglich die Echtzeitprogramme setzen das Betriebssystem MP/M voraus, da sie dessen Multitasking nutzen. Als Hauptspeicher reichen bereits 64 kBytes aus, und als Massenspeicher dienen zwei Floppy Disk Laufwerke mit insgesamt knapp einem MByte. In einer solchen Konfiguration ist allerdings ein häufiger Wechsel der Disketten unumgänglich. Da sehr viele Resultate eine graphische Ausgabe erfordern, sollte Wert auf ein Sichtgerät mit Feingrafik gelegt werden, obgleich sie nicht zwingend notwendig ist. Außerdem sind natürlich ein Prozeßanschluß über AD- und DA-Umsetzer und digitale Signalanschlüsse vorgesehen.

Hauptspeicher (64 kB)

0000H	32 kB Steuerprogramme und Reglerentwürfe auch während der Regelung austauschbar
8000H	16 kB Echtzeitprogramme und Parameter der Regelung
C000H	16 kB Betriebssystem MP/M
FFFFH	

Bild 3.2.1 Speicheraufteilung für ein minimales MP/M Betriebssystem

Unter dem Betriebssystem MP/M wurde eine Aufteilung des Speichers entsprechend Bild 3.2.1 vorgenommen. Der Gesamtspeicher von 64 kBytes teilt sich in drei Teile auf. Im obersten Bereich von 16 kBytes befindet

sich das Betriebssystem selbst. Der mittlere Bereich umfaßt ebenfalls 16 kBytes und ist für Echtzeitprogramme reserviert. Da Echtzeitprogramme keine Textein- und -ausgaben abwickeln, welche sehr speicherplatzzehrend sind, ist dieser Platz durchaus für alle normalen Aufgaben ausreichend. Im Falle der in Kapitel 7.2. beschriebenen blockorientierten Regelung reicht er im Mittel für weit mehr als 100 Blöcke, und bei Zustandsreglern mit Beobachter, wie sie in Kapitel 7.3 beschrieben sind, reicht er bis zu 8. Ordnung. Weil bei den Zustandsreglern eine Reihe von zusätzlichen Matrizen für Störgrößenaufschaltungen, PI-Zustandsregelung usw. vorgesehen sind, welche nicht bei jeder Anwendung benötigt werden, kann zusätzlicher Platz für dimensionsmäßig noch größere Regelungen durch Verzicht auf einige Optionen erreicht werden.

Alle Steuerprogramme können ausnahmslos verlassen werden, obgleich die Regelung weiterarbeitet. Deshalb müssen alle Parameter der Regler in den Echtzeitprogrammen aufbewahrt werden. Lagen sie innerhalb des Speicherbereiches der Steuerprogramme vor, so würde der Übergang zu einem Entwurfsprogramm diese Daten zerstören und die Regelung unmöglich machen. Da die Programmiersprache Pascal, anders als FORTRAN keine Common-Bereiche zuläßt, ist ein Echtzeitcommon nicht vorgesehen.

3. 2. 2. Intertaskkommunikation

Ein grundsätzliches Problem bei der Programmierung von Echtzeitprogrammen, die mit anderen Programmen Daten austauschen müssen, liegt in der Aufbewahrung dieser Daten. Ein bei vielen Prozeßrechnern gangener Weg, ist die Definition eines speziellen COMMON-

Bereiches, auf den die verschiedenen Programme gemeinsamen Zugriff haben. Eine solche Struktur eines "shared memory" ist vor allem bei einer Programmierung mit der Sprache FORTRAN leicht einzusetzen. Eine grundsätzlich andere Möglichkeit der Kommunikation mehrerer Programme miteinander ist die Verwendung von Mailboxes. Diese werden vom Betriebssystem verwaltet und können zugleich auch mit dem Starten von Programmen verknüpft werden. Der grundlegende Unterschied besteht in der Notwendigkeit von Datenübertragungen innerhalb des Rechners bei Mailboxes, welche im Falle des "shared memory" nicht erforderlich sind. Prinzipiell gleich ist beiden Methoden die Notwendigkeit, die Konsistenz aller Parameter sicherzustellen. In jedem Fall ist durch die Programme auszuschließen, daß bei Änderungen irgendwelcher Daten das Echtzeitprogramm zu einem Zeitpunkt arbeitet, zu dem erst ein Teil der neuen Daten vorliegt.

Bei der Verwendung von Pascal gibt es keinen Echtzeit-COMMON. Trotzdem kann eine gemeinsame Verwendung von Parametern realisiert werden, indem man den Zugriff auf Daten eines anderen Programms über Zeiger behandelt. Pascal stellt hierfür bereits in seiner Sprachdefinition ein Werkzeug zur Verfügung. Die Zeiger sind dann zu Beginn des Programms auszutauschen. Solange sich die Position einzelner Variablen im Rechner nicht dynamisch verändern kann, sind die einmal übertragenen Zeiger unverändert gültig. Die Wahl möglichst gleicher Namen der Variablen im Echtzeitprogramm und im Steuerprogramm erlaubt nicht zuletzt, auch in den Programmquellen eine übersichtliche Darstellung.

3. 3. Dateienmanagement

Eine der wichtigsten Schnittstellen zwischen einzelnen Programmen sind die von ihnen angelegten Dateien. Sie sind das eigentliche Werkzeug, mit dem ein entworfener Regler vom Entwurfsprogramm zum Regelprogramm übertragen wird. Der Entwurf dieser Schnittstelle wird also auf alle Programme einen erheblichen Einfluß haben.

Hierzu wurden Programmmodule erstellt, die zur Anlage von Standarddateien eingesetzt werden und die Speicherung der verschiedenen Informationen auf dem Massenspeicher optimieren können. Die in der Regelungstechnik verwendeten Dateien werden meist aus sehr verschiedenen Datentypen aufgebaut sein [H8]. So enthält eine Standarddatei für Übertragungsfunktionen im wesentlichen die Koeffizienten des Zähler- und des Nennerpolynoms - sie sind Realzahlen. Zugleich aber müssen die jeweiligen Polynomordnungen - als Integerwerte - aufgezeichnet werden. In jedem Fall muß noch die Totzeit des Systems angegeben werden. Sie könnte eine Realzahl oder eine Integerzahl sein, abhängig davon, ob das System kontinuierlich oder diskret beschrieben wurde. Eine weitere wichtige Information ist die Abtastzeit, die im Falle eines kontinuierlichen Systems Null ist. Um die Bedeutung und den Einsatzzweck einer Datei festzuhalten, sind textliche Informationen ein ganz wesentlicher Bestandteil einer jeden Datei. Manchmal mag man sie automatisiert erzeugen können - etwa durch Angabe eines Entwurfsalgorithmus und der Datei, die die Regelstreckenbeschreibung enthielt. Sehr oft ist der Benutzer gefordert, sinnvolle Erläuterungen zu nennen, die ihm auch nach Monaten noch hinreichend aufschlußreich erscheinen.

Zur Realisierung solcher Standarddateien wurde ein speicherplatzoptimierendes Interface programmiert, welches von allen Programmen genutzt wird. Es besteht

aus einem jeweils gleichen Dateienkopf, in dem der Typ der Datei, der Name des erzeugenden Programms, ein freier Kommentar und andere allgemeine Daten enthalten sind, und einem zweiten Teil, der die dateispezifischen Informationen umfaßt. Letztere sind beispielsweise Ordnungen der Polynome in einer Übertragungsfunktion und die zugehörigen Koeffizienten, oder aber die Dimensionen von Matrizen und die Elemente der Matrizen selbst.

Bei der häufigen Verwendung von Dateien hat sich gezeigt, daß solche Schnittstellen die Programmierung wesentlich vereinfachen können und zugleich ein gutes Mittel zur Vereinheitlichung darstellen.

3. 4. Prozeßankopplung

Die Kopplung eines Prozesses an einen Rechner wird in den meisten Fällen über Analog-Digital- und Digital-Analog-Wandler erfolgen. Der Zugang zu solchen Einrichtungen ist mit den normalen Werkzeugen einer höheren Programmiersprache nicht möglich, da Zugriffe auf absolute Adressen des IO-Bereiches nicht unterstützt werden. Teilweise enthalten alle Compiler Ergänzungen zum normalen Standard, die solche Funktionen zulassen. Wenn diese genutzt werden, dann sind die Programmteile, die auf Wandler zugreifen, natürlich nicht portabel. Allein schon die Tatsache, daß verschiedene Adressen verwendet werden können oder daß die Ansprache der Wandler unterschiedlich sein kann, verhindert dies. Zweckmäßiger ist es, für das Einlesen von Werten von den AD-Kanälen und das Ausgeben von Werten über die DA-Kanäle Unterprogramme zu schreiben, die den jeweiligen Anwendungsprogrammen hinzugefügt werden [H2]. Ob hierfür die Erweiterungen der Compiler genutzt werden können oder ob die Unter-

programme in Assembler programmiert werden, ist ohne Belang. Im Falle einer Softwareportierung können diese kleinen Programme sehr leicht neu erstellt werden. Bei einer Vielzahl von Versuchen haben sich Wandler mit einer Genauigkeit von 12 Bit bewährt.

In einigen Anwendungen müssen zur Meldung von Endebegrenzungen, Zuständen oder ähnlichem neben analogen Signalen auch einige binäre Informationen verarbeitet werden. Grundsätzlich gilt hier das für die analogen Datenpfade Gesagte. Aufgrund der sehr unterschiedlichen Art der digitalen Ein- und Ausgänge und ihrer ebenfalls sehr verschiedenen Ansprache müssen hier aber jeweils passende Unterprogramme erstellt werden.

Eine dritte, sehr spezialisierte Form der Prozeßankopplung liegt bei der Behandlung von Interrupts vor. Auch hier müssen jeweils für die Anwendung gesondert Serviceroutinen erstellt werden. Die hardwaremäßige Realisierung der Unterbrechungsanforderungen wird sich überwiegend in diesen Programnteilen widerspiegeln. Oberhalb dieser Softwareebene kann dann ein ganz normales und auch portables Programm vorliegen.

4. Dienstprogramme für regelungstechnische Aufgaben

In den vorangegangenen Kapiteln wurde die Umgebung dargestellt, soweit sie heute als Teile der Systemsoftware von Mikrorechnern anzutreffen ist, bzw. wie sie zunächst ergänzt werden mußte - Anschluß an die Echtzeit, Ansprache der Prozeßperipherie usw. Hierauf aufbauend, wurde die eigentliche Aufgabenstellung regelungstechnischer Programme realisiert. Obwohl inzwischen eine erhebliche Menge an unterstützender Software vorliegt, welche die wesentlichen Teilbereiche der Analyse, Synthese, Simulation und Regelung unter Echtzeitbedingungen abdeckt, bleibt das Gesamtsystem seiner Konstruktion als offenes System nach immer unvollständig und wird auch in Zukunft Erweiterungen in nahezu beliebiger Weise ermöglichen. Die nachfolgende Übersicht zeigt in sehr knapper Form die verfügbaren Möglichkeiten, wie sie im Augenblick vorliegen. In der Mehrzahl der angeführten Bereiche gibt es verschiedene Verfahren oder Variationen, die wahlweise benutzt werden können, oder es sind eine Anzahl von Optionen vorhanden, welche es erlauben, sich den aktuellen Erfordernissen einer Aufgabe anzupassen.

Zum Zwecke der Analyse :

- Meßwerterfassung und Auswertung,
- Identifikation,
- Korrelationen,
- Ermittlung von Systemeigenschaften wie Pol- und Nullstellenlage, Verstärkungsfaktoren, Steuer- und Beobachtbarkeit usw.,
- Simulation von Systemen mit verschiedenen Methoden.

Zum Entwurf von Regelungen :

- algebraische Reglerentwürfe mit Frequenzbereichsmethoden,
- Reglerentwürfe im Zustandsraum nach verschiedenen Entwurfsalgorithmen,
- Beobachterentwürfe,
- Filterentwürfe,
- Regelkreissimulation,
- Analyse von Regelkreiseigenschaften.

"on-line" Regelprogramme :

- diskrete und quasikontinuierliche Regler als dynamische Rückführung durch Übertragungsfunktionen mit und ohne Vorfilter,
- blockorientierte Regelung,
- Zustandsregler mit und ohne Beobachter, mit und ohne integralen Anteil und weiteren optionalen Aufschaltungen.

Nachfolgend werden zunächst drei sehr allgemeine Dienstprogramme behandelt, die Funktionen enthalten, welche innerhalb eines ausgebauten Softwaresystems ständig und in wechselndem Zusammenhang verwendet werden. Funktionen dieser Art sollten nicht ständig neu programmiert werden, sondern an einer möglichst zentralen Stelle vorliegen.

In den anschließenden Kapiteln 5. bis 8. erfolgt dann eine Darstellung unter Berücksichtigung einer thematischen Gliederung.

4. 1. Handhabung von Matrizen und Systemen im Zustandsraum

Zu den wichtigsten Beschreibungsformen dynamischer Systeme gehört die Zustandsraumdarstellung, welche durch drei Matrizen A, B und C das Verhalten eines Systems definiert. Diese Matrizen müssen passende Dimensionen besitzen

<u>A</u>	$n \times n$	Systemmatrix
<u>B</u>	$n \times r$	Eingangs- oder Steuermatrix mit r der Anzahl von Eingängen
<u>C</u>	$m \times n$	Ausgangs-, Meß- oder Beobachtungsmatrix mit m der Anzahl von Ausgängen.

Bei sprungfähigen Systemen kommt hierzu noch eine Durchgangsmatrix D, die im weiteren aber nicht berücksichtigt werden soll.

Die Behandlung von Systemen im Zustandsraum, die Berechnung ihrer Eigenwerte, Transformationen der Darstellung, Regler- und Beobachterentwürfe usw. sind nur mithilfe eines Digitalrechners durchführbar - wenn man von einfachsten Beispielen mit niedriger Dimension absieht [U4]. Um die Zustandsraumdarstellung des jeweils zu behandelnden Systems erstmalig in einer rechnergeeigneten Form einzugeben und um erste grundlegende Untersuchungen des Systems durchzuführen, gibt es ein umfangreiches Programm. Es unterstützt neben der Betrachtung von Zustandsraumdarstellungen auch eine ganze Reihe von Matrizenoperationen [Z3]. In Grenzen stellt es daher einen "Taschenrechner" der Matrizenrechnung dar. Matrizen können z.B. addiert, subtrahiert, multipliziert oder transponiert werden. Gemeinsam mit der Möglichkeit, einzelne Matrizen in Dateien abzulegen oder sie aus solchen zu lesen, sind auch kompliziertere Matrizenrechnungen, die man kaum mehr von Hand ausführen möchte, mit einigem Komfort lösbar. Auch zur Lösung von Gleichungssystemen, zur Matrix-

inversion, zur Berechnung der Determinante einer Matrix und zur Untersuchung ihrer Definitheit steht leistungsfähige Software zur Verfügung [L3]. Hierzu werden in Pascal formulierte Versionen der bekannten LINPACK-Routinen eingesetzt, die bestmögliche Resultate im Rahmen der durch die Realwortlänge gegebenen Genauigkeit erwarten lassen [D2]. Die Realwortlänge ist von der jeweiligen Pascal-Implementierung abhängig. Beispielsrechnungen mit einer Reihe von Testmatrizen haben gezeigt, daß auch mit einer Darstellung REAL*4 (IEEE single precision) gute Ergebnisse erwartet werden können [G4, H9].

Faßt man die drei Matrizen A, B und C bei passenden Dimensionen zusammen, so erhält man eine Zustandsraumdarstellung. Diese bilden die Grundlage für die Handhabung und Untersuchung von dynamischen Systemen. Erste Grundoperationen sind Veränderungen in der Reihenfolge der Elemente des Zustandsvektors oder das Vertauschen, Hinzufügen oder Wegnehmen von einzelnen Eingängen oder Ausgängen. Die erstgenannte Operation, das Umsortieren von Zustandsgrößen, bewirkt keine Veränderung am Eingangs-Ausgangsverhalten des Systems, da sie eine Ähnlichkeitstransformation ist. Weitere elementare Ähnlichkeitstransformationen erlauben das Balancieren oder Skalieren von Zeilen und Spalten und das Erzeugen von Null- oder Einselementen an bestimmten Stellen in den Matrizen. Eine Hauptanwendung finden solche Operationen vielfach ausgeführt bei Untersuchungen von Systemeigenschaften, wie sie bei Rosenbrock [R2] angegeben sind. Alle im Verlauf der Bearbeitung durchgeführten Elementartransformationen können in gesondert gespeicherten Transformationsmatrizen zusammengefaßt werden und stehen für spätere Rücktransformationen zur Verfügung.

Ein ganz wesentlicher weiterer Programmteil berechnet die Eigenwerte der Systemmatrix. Auch hier wurden in Pascal formulierte Versionen der Unterprogramme der EISPACK-Routinen [G1, L3, S1] bzw. der

ursprünglich in Algol geschriebenen Programme eingesetzt [W5]. Auch sie bedingen Resultate mit recht guter Genauigkeit unter Berücksichtigung der jeweils implementierten Zahlendarstellung.

Der Übergang von einer kontinuierlichen zu einer diskreten Darstellung und ihre Umkehrung runden das Programmpaket ab. Sie erfolgen in klassischer Weise mithilfe der Fundamentalmatrix [A1, U4].

Eine weitere Grundfunktion erlaubt das Zusammenschalten zweier Systeme. Hierbei kann je nach den Erfordernissen zwischen einem Parallelschalten, einem Zusammenschalten in Reihe oder einer negativen Rückführung über ein zweites System gewählt werden. Als Resultat ergibt sich ein neues System, dessen Ordnung gleich der Summe der Ordnungen der beiden zusammenschalteten Systeme ist. Für weitere Analysen kann es wie andere Systeme im Zustandsraum auch weiterverwendet werden.

4. 2. Handhabung von Übertragungsfunktionen

Ähnlich dem in Kapitel 4.1. beschriebenen Programm zur Handhabung von Systemen im Zustandsraum, existiert auch für kontinuierliche und diskrete Übertragungsfunktionen ein Programm, welches die wichtigsten Grundoperationen und Untersuchungen in interaktiver Form benutzbar macht. Für viele andere Programme des Systems stellt es zugleich die erste Definitionsmöglichkeit von Übertragungsfunktionen dar. Die Eingabe der Übertragungsfunktion entweder über die Koeffizienten oder über die Pole und Nullstellen mit einem zusätzlichen Verstärkungsfaktor und das anschließende Ablegen der Übertragungsfunktion auf eine entsprechende Standarddatei ist eine Grundfunktion, die von anderen

Programmen vorausgesetzt wird.

Liegt nur eine Übertragungsfunktion vor, so können die kontinuierlichen oder auch diskreten Pole und Nullstellen berechnet werden [S1]. Falls rückgekoppelte Systeme untersucht werden sollen, kann man mehrere Systeme parallel oder in Reihe schalten oder aber eine negative Rückführung wahlweise über ein zweites System oder auch eine konstante Verstärkung hinzufügen. Dabei sind sowohl bei kontinuierlichen als auch bei diskreten Systemen immer Totzeitanteile erlaubt, sofern die Übertragungsfunktion als rationale Funktion und einem einzelnen multiplikativen Totzeitanteil dargestellt werden kann. Alle verwendeten Übertragungsfunktionen können von Dateien gelesen und alle berechneten Übertragungsfunktionen für die Verwendung in anderen Programmen auf entsprechende Dateien abgelegt werden.

4. 3. Meßwerterfassung

Eine ganz wichtige Aufgabe für ein regelungstechnisches Softwaresystem ist ein Programm zur Erfassung und Speicherung von Prozeßsignalen. Es liefert für viele weitere Arbeiten die dringend nötige Grundinformation. So sind etwa Modellbildungen durch Identifikation nur mit Prozeßsignalen möglich.

Erfaßt werden können analog gemessene Signale mit einer einstellbaren Abtastzeit [A6]. Der Beginn einer Messung kann im einfachsten Fall von Hand am Rechner selbst erfolgen. In den weitaus häufigeren Fällen wird man aber einen automatischen Aufzeichnungsbeginn benötigen, da eine sinnvolle Messung erst mit einem Ereignis wie etwa einem veränderten Sollwert oder dem Eintritt einer Störung beginnen kann. Der Aufzeichnungsbeginn (Triggerung) wird dann mit einer

positiven oder negativen Flanke eines gemessenen Signals erfolgen.

Zu jedem Abtastzeitpunkt werden ein oder mehrere Werte von den Wandlern gelesen und nacheinander auf Signaldateien abgelegt. Normalerweise geschieht dies bereits in der Form von Realzahlenwerten, die auf den Bereich von ± 10 Volt mit ± 1.0 normiert sind. Bei besonders schnell anfallenden Signalen kann eine Speicherung auch unmittelbar im Wandlerformat erfolgen. Vor einer Weiterverarbeitung müssen diese Signaldateien meist umgesetzt werden in die standardisierte Form.

Innerhalb des interaktiven Programms können die Anzahl der Meßkanäle, die Zuordnung zu den einzelnen Wandlern, die Triggerbedingungen und die Form der Datenhaltung eingestellt, verändert oder auch während einer Messung angezeigt werden. Besonders wichtig für die spätere Weiterverwendung der Signale sind Angaben, die die Art der Signale betreffen und textliche Bezeichnungen jedes einzelnen Signals und der gesamten Messung. Die Signalaufzeichnung erfolgt durch aufeinanderfolgende Werte, denen nicht mehr angesehen werden kann, ob es sich um kontinuierliche oder diskrete Signale handelt. Ohne eine ausreichende Bezeichnung der Signale und Messungen können auch schon kleinere Meßwerterfassungen nicht mehr sortiert gehalten werden.

Die zweite wichtige Aufgabe einer Meßwerterfassung besteht in der graphischen Aufbereitung der vorliegenden Signale. Sie kann wahlweise auf verschiedenen Ausgabegeräten erfolgen. Zu einer visuellen Inspektion dient der Bildschirm, da Signale hierauf schnell und problemlos dargestellt werden können. Im simpelsten Fall kann als Ausgabegerät sogar ein Terminal ohne eigene Graphikunterstützung verwendet werden. Die Ausgabe erfolgt dann als "Sternchengraphik". Zum Erkennen des groben Verlaufs eines Signals reicht eine solche Ausgabeform aus, nützlicher aber ist eine Feingraphik auf einem hierfür

vorbereiteten Terminal, welches in den aktuellen Versionen eines der Hewlett-Packard 264x Sichtgeräte sein kann. Als Alternative mit wegnehmbaren graphischen Darstellungen dient ein Plotter.

Die graphische Darstellung von Signalen kann auch von anderen Programmen genutzt werden, die keine eigene Graphik besitzen. Zur Auswertung wichtig ist eine klare Unterscheidung von Signalen, die durch Abtasten einer kontinuierlichen Größe entstanden sind, von solchen, die eine diskrete Größe darstellen. Alle Programme, die Signale erzeugen oder ermitteln, tragen in den jeweiligen Signaldateien entsprechende Informationen ein. Meist geschieht dies automatisch, da durch die jeweilige Umgebung klar ersichtlich ist, ob ein aufgezeichnetes Signal kontinuierlich oder diskret ist. Als Sonderfall dient die gleiche Graphik auch zur Darstellung von Korrelationsfunktionen, weil diese ebenfalls in Signaldateien abgelegt werden.

Da die Zahl der zur Zeit angeschlossenen Analog-Digital-Wandler über zwei Multiplexer auf 16 Kanäle begrenzt ist, werden auch maximal 16 Signale zu jedem Abtastzeitpunkt erfaßt. Entsprechend der Natur der Signale kann man sie für eine spätere Verarbeitung auch gemischt als kontinuierlich oder diskret kennzeichnen und mit jeweils einem eigenen Namen versehen. Die Anzahl der Abtastschritte kann entweder vor Beginn der Messung festgelegt werden, oder sie ergibt sich von selbst durch eine manuelle Unterbrechung der Meßwertaufnahme durch die Anzahl der bis dahin durchgeführten Abtastschritte.

5. Identifikation

Eine wesentliche und normalerweise erste Aufgabe der Regelungstechnik ist die analytische Durchdringung des Problems. Hierzu gehört einerseits eine mathematisch-physikalische Analyse, welche ein Modell der Regelstrecke liefern kann. Ein anderer Weg wird mit den Methoden der Identifikation gegangen. Vermittels statistischer Untersuchungen kann ebenfalls Information über das dynamische Verhalten der Strecke gewonnen werden.

Das vorliegende Softwaresystem verfügt derzeit über Programme zur Korrelationsanalyse und zur Parameteridentifikation. Wenn hier Korrelationsmethoden eingesetzt werden, erhält man zunächst eine nichtparametrische Beschreibung der Regelstrecke durch Auto- und Kreuzkorrelationsfunktionen. Durch eines der Identifikationsprogramme kann eine parametrische Beschreibung durch Übertragungsfunktionen direkt ermittelt werden. Sie steht anschließend allgemein zur Verfügung und kann zur Simulation oder zum Reglerentwurf eingesetzt werden.

5.1. Korrelationsanalyse

Bereits ganz am Anfang der Entwicklung dieses Softwaresystems stehen Programme zur analytischen Behandlung von Systemen mit statistischen Methoden. Anhand gemessener Signale der Regelstrecke werden Auto- und Kreuzkorrelationen ermittelt. Gleichzeitig mit den erforderlichen Berechnungen können Testsignale als Stellgrößen durch einen digitalen Rauschgenerator erzeugt werden [S6]. Es stehen dabei mehrere verschiedenartige Impulsfolgen zur Verfügung, die

zeitlich und amplitudenmäßig variiert werden können, um den Algorithmus den speziellen Eigenschaften des Untersuchungsobjektes anzupassen. Prinzipiell sollte man versuchen mit möglichst großen Amplituden zu arbeiten, jedoch sind die hierdurch verursachten Betriebsstörungen der Anlage oft unerwünscht oder sogar unzulässig. Die Bildung der Korrelationsfunktionen kann mit verschiedenen Genauigkeiten erfolgen, um bei schnellen Systemen mit geringerer Genauigkeit Rechenzeitkonflikte zu vermeiden oder bei langsameren Systemen mit entsprechend höherer Genauigkeit arbeiten zu können.

Die erhaltenen Korrelationsfunktionen liefern nach einer Fast-Fourier-Transformation [B11, C5] oder einer numerischen Fourier-Transformation durch Knickgeradenapproximation [U5] die zugehörigen Frequenzspektren, welche zur punktweisen Berechnung des Frequenzganges des untersuchten Systems herangezogen werden [S4].

Die Berechnung von Korrelationfunktionen beliebiger Signale wurde in dem Programm KOR in Pascal formuliert. Grundlage bilden hierbei Standardsignaldaten, wie sie von anderen Programmen des Systems - insbesondere von der Meßwerterfassung - erzeugt werden können [K3]. Dialoggesteuert werden zwei der vorhandenen Signale ausgewählt. Falls es sich bei beiden um das gleiche Signal handelt, wird die Autokorrelationsfunktion berechnet

$$R_{xx}(\tau) = \frac{1}{N} \sum_{v=0}^{N-1} x(v) x(v+\tau)$$

und falls zwei verschiedene Signale ausgewählt wurden, werden sie kreuzkorreliert [L6, U5]

$$R_{xy}(\tau) = \frac{1}{N} \sum_{v=0}^{N-1} x(v) y(v+\tau)$$

Schließlich kann ein Signal daraufhin untersucht werden, ob es mit einer Wahrscheinlichkeit von 95% als weißes Rauschen angesehen werden darf. Durch einen solchen T-Test kann entschieden werden, ob das gemessene Signal als Eingangssignal einer Regelstrecke geeignet ist, d. h. ob die im System vorhandene Dynamik im gesamten interessierenden Spektrum ausreichend angeregt wird.

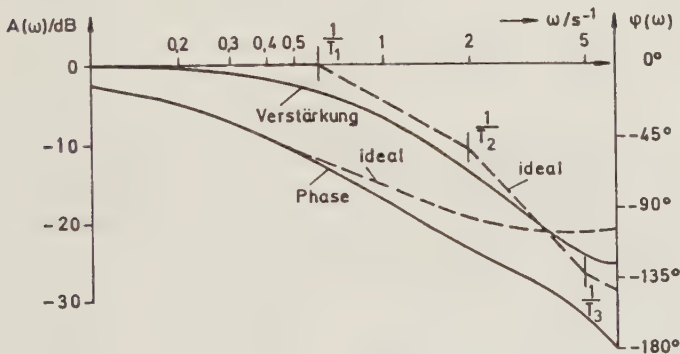


Bild 5.1.1 Bode-Diagramm aus einer Korrelation und Fast-Fourier-Transformation

Bild 5.1.1 zeigt ein Bode-Diagramm, wie es als Resultat einer Korrelation und anschließender Fast-Fourier-Transformation entsteht [S7]. Als Regelstrecke wurde ein System von 2. Ordnung als Analogrechnerschaltung realisiert. Seine Übertragungsfunktion lautete

$$G(s) = \frac{(T_3 s + 1)}{(T_1 s + 1)(T_2 s + 1)}$$

mit $T_1 = 1.6961$
 $T_2 = 0.5093$
 $T_3 = 0.2000$.

Es erfolgte eine Echtzeitkorrelation mit einer Abtastzeit von 500 msec. Das Ergebnis ist sehr gut, wenn man berücksichtigt, daß bei dieser Abtastzeit nur die recht geringe Anzahl von 128 Meßwertpaaren verwendet wurde. Der Einfluß der recht geringen Abtastzeit ist bei höheren Frequenzen vor allem im Verlauf der Phase zu erkennen.

5. 2. Parameteridentifikation

Werden gemessene Signale der Ein- und Ausgänge eines Regelsystems zur Grundlage einer Schätzung der Parameter einer Übertragungsfunktion gemacht, so gelangt man zu den Methoden der experimentellen Systemidentifikation. Seit der Mitte der sechziger Jahre wurde eine Vielzahl von Verfahren und Variationen entwickelt. Zwei dieser Verfahren - die Methode der kleinsten Quadrate ("Least-Squares"-Verfahren) und die Hilfsvariablenmethode (Instrumental Variable Verfahren) - sind durch entsprechende Programme im vorgestellten Softwaresystem zugänglich.

Üblicherweise setzen alle Identifikationsverfahren voraus, daß der Prozeß durch ein einziges Modell mit genau einem Parametersatz vollständig beschrieben werden kann. Das bedeutet, daß der untersuchte Prozeß unter allen Arbeitsbedingungen linear ist, und seine dynamischen Eigenschaften keiner Arbeitspunktabhängigkeit unterliegen. Das System wird daher durch eine Übertragungsfunktion

$$G(z) = \frac{\sum_{i=1}^n b_i z^{-i}}{1 + \sum_{i=1}^n a_i z^{-i}}$$

beschrieben.

Bei vielen realen Prozessen ist die Übertragungsfunktion vom Betriebszustand abhängig. Anstatt für den gesamten Arbeitsbereich nur eine einzige Übertragungsfunktion zu identifizieren, die im Prinzip eine Mittelung über alle Arbeitspunkte enthält, kann man für eine Anzahl von Arbeitspunkten jeweils eigene Übertragungsfunktionen vorsehen [D3]. Während der Identifikation müssen alle Arbeitspunkte ausreichend durchlaufen werden, damit von einer Umschaltung alle diese Übertragungsfunktionen geschätzt werden können. Eine solche Parallelidentifikation eines stückweise linearisierbaren Systems enthält den Standardfall eines linearen Systems als Sonderfall. Versuche an verschiedenen Testsystemen haben gezeigt, daß mit der Parallelidentifikation genauere Resultate erzielbar sind [K3].

Die Koeffizienten der oben angegebenen Übertragungsfunktion als dynamisches Modell der Regelstrecke werden zu einem Parametervektor p zusammengefaßt

$$p = [a_1 \dots a_n \ b_1 \dots b_n]^T$$

und die am Streckenein- und -ausgang abgegriffenen Signalwerte bilden zusammengefaßt einen Meßvektor m

$$m(k+1) = [-y(k) \dots -y(k+1-n) \ u(k) \dots u(k+1-n)]^T .$$

Das Ziel der Parameterschätzung soll durch Bild 5.2.1 erläutert werden. Das untersuchte System wird durch die Eingangsgröße u erregt und liefert y_s als Ausgangsgröße, der sich ein Rauschanteil r_s überlagert und damit den meßbaren Streckenausgang y bildet [U5]. Zur Identifikation wird dem untersuchten System das Modell parallelgeschaltet. Der Identifikationsalgorithmus verstellt unter Kenntnis des Eingangssignales u und des Ausgangssignales y der Strecke den Parametervektor dieses Modell solange, bis ein Fehlersignal als Differenz zwischen Regelstreckenausgang und Ausgang des Modells über ein Gütekriterium minimiert ist. Beim "Least-Squares"-Verfahren durch

$$e(k+1) = y(k+1) - \hat{m}^T(k+1)p(k)$$

und bei der Hilfsvariablenmethode durch

$$v(k+1) = y(k+1) - \hat{m}^T(k+1)p(k)$$

wird dieses Fehlersignal auf gleiche Weise gebildet.

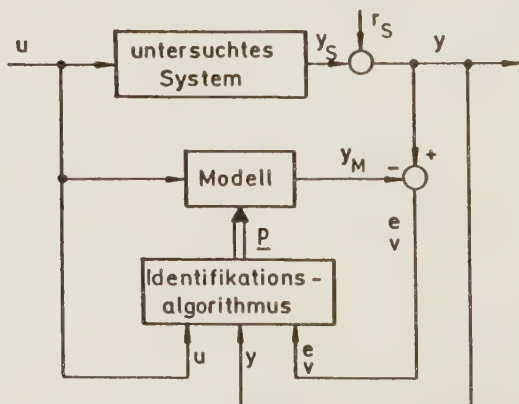


Bild 5.2.1 Prinzipdarstellung der Parameterschätzung

In der rekursiven Form des "Least-Squares"-Verfahrens ermittelt man zu jedem Abtastschritt folgende Vektoren und Matrizen aus denen des vorangegangenen Abtastschrittes und den aktuellen Ein- und Ausgangsgrößen der Regelstrecke

$$\underline{g}(k+1) = \underline{P}(k)\underline{m}(k+1)[1 + \underline{m}^T(k+1)\underline{P}(k)\underline{m}(k+1)]^{-1}$$

$$\hat{p}(k+1) = \hat{p}(k) + \underline{g}(k+1)e(k+1)$$

$$\underline{P}(k+1) = \underline{P}(k) - \underline{g}(k+1)\underline{m}^T(k+1)\underline{P}(k) \quad .$$

Falls mit mehr als nur einer Übertragungsfunktion gearbeitet wird, d.h. im Falle der Parallelidentifikation, liegen natürlich entsprechend viele Parametervektoren $\underline{p}$ vor. Die Matrix $\underline{P}$ kann entweder gemeinsam für alle Parametersätze verwendet werden, oder man gibt auch jedem Parametersatz seine eigene Matrix $\underline{P}$. Verschiedene Versuche haben ergeben, daß hiermit meist eine Verbesserung der Schätzung erreicht wird [K3]. Durch jeweils eigene $\underline{P}$ -Matrizen zu jedem Parametersatz kann ein frühzeitiges "Einschlafen" des Algorithmus vermieden werden.

Bei der Hilfsvariablen-Methode wird zusätzlich ein Parametervektor $\underline{p}_H$ eines Hilfsmodells eingeführt und mitgeschätzt. Neben dem bereits eingeführten Meßvektor $\underline{m}$ tritt für das Hilfsmodell ein entsprechender Vektor $\underline{w}$

$$\underline{w}(k+1) = [-y_H(k) \dots -y_H(k+1-n) \quad u(k) \dots u(k+1-n)]^T$$

auf, in dem die enthaltene Ausgangsgröße y_H des Hilfsmodells analog zur Bildung des Fehlersignals gebildet wird

$$y_H(k+1) = \underline{w}^T(k+1)\underline{p}_H(k+1) \quad .$$

Zusätzlich wird dieser Parametervektor durch ein diskretes, totzeitbehaftetes Verzögerungsglied geführt [17, U5, Y1]

$$\hat{p}_H(k+1) = (1-\gamma)\hat{p}_H(k) + \gamma p(k-d+1) \quad .$$

Hiermit und mit dem Fehlersignal $v(k+1)$ liegen nun alle Größen vor, um zu jedem Abtastschritt des Algorithmus die Vektoren und Matrizen

$$\underline{q}(k+1) = \underline{P}(k)\underline{w}(k+1)[1 + \underline{m}^T(k+1)\underline{P}(k)\underline{w}(k+1)]^{-1}$$

$$\hat{p}(k+1) = \hat{p}(k) + \underline{q}(k+1)v(k+1)$$

$$\underline{P}(k+1) = \underline{P}(k) - \underline{q}(k+1)\underline{m}^T(k+1)\underline{P}(k)$$

neu zu berechnen. Auch bei diesem Identifikationsverfahren ist eine Parallelidentifikation mit mehreren Übertragungsfunktionen möglich.

Ohne eine zusätzliche Gewichtung der Schätzergebnisse kommt es meist zu sehr schnellen, starken Verkleinerungen der Elemente der $\underline{P}$ -Matrix. Es ist daher von Vorteil, dies durch die Einführung von üblicherweise konstanten Gewichten ρ bei der Bildung dieser Matrix zu beeinflussen [U5]

$$\underline{P}(k+1) = \frac{1}{\rho} [\underline{P}(k) - \underline{q}(k+1)\underline{m}^T(k+1)\underline{P}(k)] \quad .$$

Die so veränderte Berechnungsvorschrift gilt sowohl für das rekursive "Least-Squares"-Verfahren als auch für die Hilfsvariablenmethode. Die Gewichtung ρ liegt zumeist in einem Bereich von 0,95 bis 0,99 und muß den jeweiligen Verhältnissen angepaßt werden. Die Auswahl eines zweckmäßigen Gewichtungsfaktors erfolgt durch Versuche so, daß die Konvergenz der geschätzten

Parameter befriedigend ist.

Beide Identifikationsverfahren liegen in zwei verschiedenen Versionen vor, die entweder in einer "off-line"-Version arbeiten und hierzu gemessene Signale einer Signaldatei verwenden oder im "on-line"-Betrieb selbst diese Messungen mit einer wählbaren Abtastrate am laufenden Prozeß durchführen. Das "on-line"-Verfahren besitzt naturgemäß den Vorteil einer ständigen Information über den augenblicklichen Stand der Identifikation. Die Verwendung von exakt den gleichen Signalen für eine Reihe von Identifikationsversuchen mit unterschiedlichen Verfahren, Anfangsbedingungen und Voreinstellungen oder bei einer veränderten Anzahl paralleler Schätzungen ist natürlich nur durch "off-line"-Identifikation zu erzielen. Dies gilt insbesondere dann, wenn die Resultate eines Identifikationslaufes dazu verwendet werden sollen, die Einstellwerte für einen nachfolgenden erneuten Identifikationlauf festzulegen.

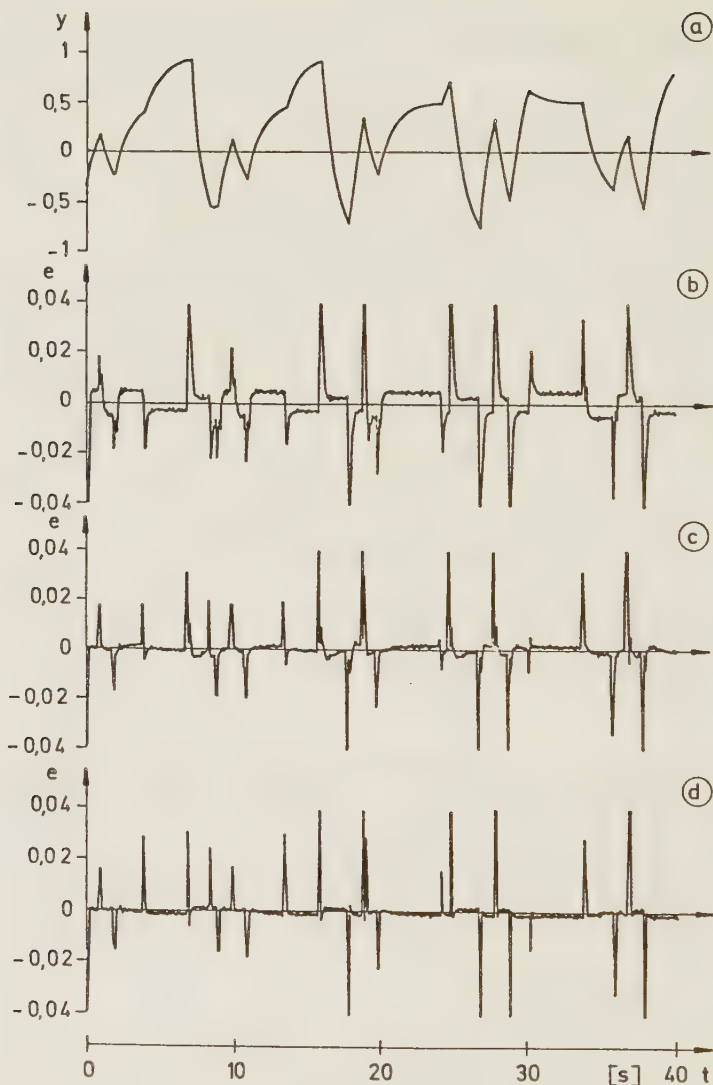
Zur näheren Untersuchung wurde ein Beispiel 2. Ordnung gewählt [K3]. Es handelt sich hierbei um ein PT₂-System mit zwei Arbeitspunkten, die sich in ihren Verstärkungsfaktoren unterscheiden. Mit einer Abtastzeit von 0,1 Sekunden berechnen sich die beiden diskreten Übertragungsfunktionen des Prozesses im

1. Arbeitspunkt zu

$$G_1(z) = \frac{0,01775 z^{-1} + 0,01233 z^{-2}}{1 - 1,2727 z^{-1} + 0,3328 z^{-2}}$$

und im 2. Arbeitspunkt zu

$$G_2(z) = \frac{0,03550 z^{-1} + 0,02465 z^{-2}}{1 - 1,2727 z^{-1} + 0,3328 z^{-2}} .$$



**Bild 5.2.2 Fehler der Einschrittvorhersage bei
Identifikation mit**

- a) gemessenes Ausgangssignal
- b) 1 Parametervektor
- c) 2 Parametervektoren, 1 P-Matrix
- d) 2 Parametervektoren, 2 P-Matrizen

Die Umschaltung zwischen den beiden Arbeitspunkten dieser Regelstreckenbeschreibung erfolgte durch ein PRBS-Signal etwa alle 40 Meßwerte. Es wurden Parameterschätzungen mit nur einem Parametersatz und als Parallelidentifikation mit zwei Parametersätzen und einer oder zwei Kovarianzmatrizen durchgeführt. Die Wichtungsfaktoren wurden jeweils durch Versuch günstig eingestellt. Bild 5.2.2 zeigt den Fehler der Einzschrittvorhersage für die verschiedenen Identifikationen unter Verwendung des "Least-Squares"-Verfahren. Bei nur einem Parametersatz entsteht ein größerer Simulationsfehler als bei Identifikation mit zwei Parametersätzen. Für eine Signalvorhersage ist der Unterschied zwischen nur einer P -Matrix und zwei getrennten P -Matrizen für die beiden Parametervektoren nur gering. Vergleicht man aber die aus den geschätzten Parametern berechneten Verläufe der Verstärkung, dann erkennt man geringere zeitliche Schwankungen, wenn zwei P -Matrizen benutzt werden - Bild 5.2.3. Bei einer Identifikation mit nur einem Parametervektor ergäbe sich ein "mittlerer" Verstärkungsfaktor, dessen Wert in stärkerem Maße von der Folge der durchlaufenen Arbeitspunkte einerseits und von dem Wichtungsfaktor und dem Abbau der P -Matrix andererseits abhängen wird.

Zwei weitere Identifikationsbeispiele ergaben vergleichbare Resultate [K3]. Bei einem labormäßigen Motor-Generator-System erzielte eine Identifikation mit drei Parametersätzen deutlich geringere Fehler der Einzschrittvorhersage als eine solche mit nur einem Parametersatz.

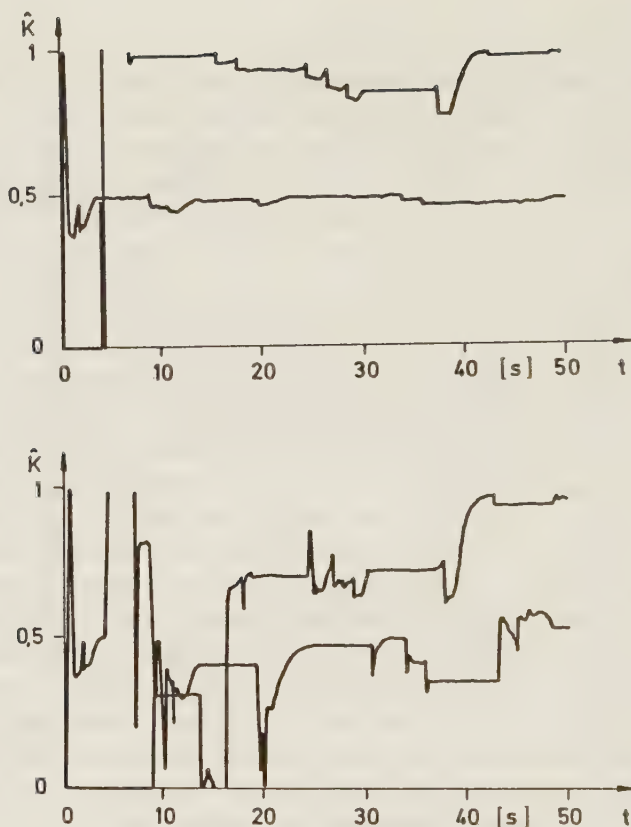


Bild 5.2.3 Verlauf der geschätzten Verstärkungsfaktoren
in beiden Arbeitspunkten
oben 1 P-Matrix, unten 2 P-Matrizen

Messungen wurden auch an einem Laborprozeß der Firma Krupp-Polysius vorgenommen. Dieser Prozeß stellt eine modellhafte Nachbildung eines Teilprozesses von Zementmühlen dar und wird normalerweise zur Beurteilung von Reglereinstellungen zur Temperaturregelung verwendet. Mit fünf Parametersätzen statt einem erhält man auch hier bessere Resultate. Der Unterschied zwischen nur einer P-Matrix und fünf P-Matrizen ist allerdings hier nicht sehr groß. Die Standardabweichung des Fehlers der Einschrittvorhersage verringert sich mit fünf P-Matrizen auf etwa die Hälfte bis ein Drittel

gegenüber einer Identifikation mit fünf Arbeitspunkten aber nur einer P-Matrix. Da dieses System durch viele Arbeitspunkte beschrieben werden mußte, um seiner verhältnismäßig starken Nichtlinearität gerecht zu werden, mag man zumindest in diesem Fall mit nur einer P-Matrix auskommen, weil die erzielbare Verbesserung den Mehraufwand kaum rechtfertigt.

6. Rechnerunterstützter Reglerentwurf

Der Entwurf eines Reglers stellt eine häufig wiederkehrende Aufgabe dar. Die eingesetzten Algorithmen sind oftmals für eine Berechnung von Hand viel zu aufwendig. Insbesondere bei höheren Ordnungen der Regelstrecke kommt dieser Weg nicht mehr in Betracht. Zweckmäßigerweise verwendet man dann ein Rechenprogramm, welches die gewünschten Resultate liefert und zugleich Randbedingungen und Einschränkungen des Verfahrens automatisch berücksichtigt. Weiterhin kann man die Ergebnisse in eine Form bringen, die ihre unmittelbare Weiterverwendung in Echtzeitprogrammen erlaubt. Hierin zeigt sich auch der eigentliche Grund, warum solche Programme unmittelbar am Einsatzort des Reglers verfügbar sein müssen. Da ein Mikrorechner-system verhältnismäßig leicht an den Ort der Regelstrecke zu bringen ist, müssen dann die Entwurfsalgorithmen für Regler ebenfalls dort vorhanden sein. Erst wenn die Rechenzeiten so groß werden, daß kleine Rechner überfordert sind, sollte man größere aber weiter entfernt aufgestellte Rechner benutzen. Die vielen Entwurfsprogramme des vorgestellten Systems haben eindeutig gezeigt, daß Mikrorechner problemlos in der Lage sind, diese Aufgabe zu erfüllen.

6. 1. Dynamische Reglerentwürfe

Eingrößenregler gehören zu den am häufigsten verwendeten Reglern. Zur Regelung von Mehrgrößensystemen können mehrere Eingrößenregler zusammengesaltet werden. Ganz typisch für eine solche Regelung sind Kaskadenregelkreise. Allerdings sind im Falle der Mehrgrößregelung besondere Maßnahmen zur Sicherung der Stabilität unumgänglich.

Um Eingrößenregler zu entwerfen, sind im Laufe der Jahre eine Vielzahl von Vorschlägen gemacht worden. Hieraus wurden einige ausgewählt und zu einem Programmpaket zusammengefaßt [A3]. Beginnend mit einem Menue werden 10 Basisentwürfe, teilweise mit einer Reihe weiterer Variationen, herausgestellt. Allen gemeinsam ist eine mathematische Behandlung im Frequenzbereich. Sie liefern als Ergebnis des Entwurfs kontinuierliche oder diskrete Übertragungsfunktionen eines Reglers und erfordern als Eingangsdaten ebenfalls kontinuierliche oder diskrete Übertragungsfunktionen einer Regelstrecke. Neben Regelstreckenmodellen aus einem Verzögerungsglied 1. Ordnung mit einer Totzeit oder linearen Tiefpaßregelstrecken werden von einigen Entwürfen auch allgemeine diskrete Übertragungsfunktionen verwendet. Als Entwurfsmethoden stehen zur Verfügung :

- diskrete PID-Regler aufbauend auf verschiedenen kontinuierlichen Entwürfen
 - Einstellregeln nach Ziegler-Nichols,
 - Einstellregeln nach Takahashi,
- Kompensationsalgorithmen für PT_1T_t -Regelstrecken
 - Dahlin-Algorithmus,
 - Proportional-(Integral)-Prädiktor-Regler,
 - Stellgrößenvorgabe- (SV-) Algorithmus,
- diskrete PID-Regler nach der Methode der Doppelverhältnisse und des Betragsoptimum,
- diskreter Kompensationsalgorithmus,
- Regler für endliche und minimale Einstellzeit für Führungs- und Störverhalten.

In aller Regel stehen für ein einzelnes Problem verschiedene Entwurfsverfahren zur Verfügung, sodaß etwa in einer Simulation das Verhalten der Regelstrecke

mit verschiedenen Reglern beurteilt werden kann.

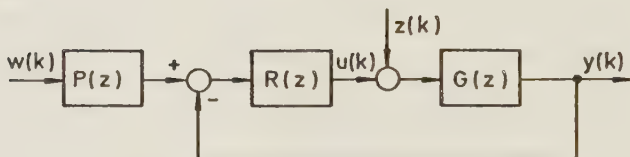


Bild 6.1.1 Regelkreis aus Regelstrecke, Regler und Vorfilter

Ausgangspunkt ist in allen Fällen zunächst eine Regelkreisstruktur wie in Bild 6.1.1 angegeben. Sie besteht aus der Regelstrecke $G(z)$ einem Regler $R(z)$ und einem Vorfilter $P(z)$. Die Aufteilung der Regelung auf einen Regler und ein Vorfilter erlaubt es, sowohl das Führungsverhalten - mittels der Übertragungsfunktion $R(z)$ - als auch das Störverhalten - mittels der Übertragungsfunktion $P(z)$ - getrennt zu entwerfen. Ohne das Vorfilter müßte versucht werden, beide Verhalten zugleich mit nur einer Reglerübertragungsfunktion zu beeinflussen. Obgleich diese einfachste Struktur sehr häufig eingesetzt wird, kann sie nur einen Kompromiß für Führungs- und Störverhalten liefern. Aber auch bei Verwenden eines Vorfilters ergeben sich gelegentlich Probleme. Häufig wird das Vorfilter einen Anteil besitzen, der unerwünschte Pole oder Nullstellen des geschlossenen Kreises aus Regelstrecke und Regler $R(z)$ kompensiert. Im Falle einer Kompensation einer Nullstelle des Reglers $R(z)$, tritt die zugehörige Wurzel als Polstelle des Vorfilters im Nenner der Übertragungsfunktion $P(z)$ auf. Liegt diese Polstelle außerhalb oder auf dem Einheitskreises, so ist das Vorfilter instabil. Im Gegensatz zum Regler $R(z)$, der durchaus instabile Anteile besitzen darf, ist ein instabiles Vorfilter unzulässig. Diesem Umstand kann jedoch auf einfache Weise abgeholfen werden, indem man die instabile Wurzel im Vorfilter und im Regler herauskürzt und im Zähler eines zusätzlichen Filters $Q(z)$ im

Rückführzweig vorsieht. Man gelangt damit zu einer Regelkreisstruktur wie in Bild 6.1.2, die ohne die Gefahr einer numerischen Instabilität das gleiche Übertragungsverhalten realisiert.

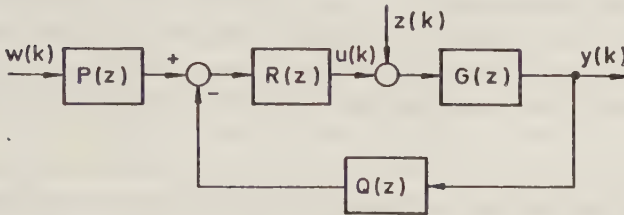


Bild 6.1.2 Regelkreis bestehend aus Regelstrecke, Regler, Vorfilter und einem Filter im Rückkopplungspfad

Als PI- oder PID-Regler sind betragsoptimale Regler in der Antriebstechnik von großer Bedeutung und werden dort vielfach verwendet [B7]. Neben einer ganzen Reihe von Ansätzen, die Einstellparameter eines kontinuierlichen PID-Reglers zu berechnen [F4, F5, F7, I1], gibt es einen Vorschlag von Strejc [S3], der die direkte Berechnung der Parameter eines diskreten PI- oder PID-Reglers ausgehend von einer diskreten Streckenbeschreibung erlaubt. Unter Zugrundelegen einer Regelstrecke $G(z)$ und eines Reglers $R(z)$ erhält man die Koeffizienten der Reglerübertragungsfunktion sowohl für PI- als auch für PID-Regler aus einfachen Gleichungssystemen [A3, S3].

Ausgehend von der Führungsübertragungsfunktion des geschlossenen Regelkreises kann man die Koeffizienten des charakteristischen Polynoms bestimmten Bedingungen unterwerfen, die man als Doppelverhältnisse bezeichnet [N1]. Diese Koeffizienten enthalten sowohl die festen Parameter der Regelstrecke als auch die gesuchten des Reglers. Durch sukzessive Berechnung läßt sich der Regler ermitteln. Nach Tröndle [T2] kann der Entwurf auf diskrete Regler erweitert werden durch die Approxi-

mation eines Haltegliedes 0. Ordnung mit einem PT_1 -Glied und der Transformation durch die Tustinformel. Tröndle bezeichnet die nach seiner Methode entworfenen Regler als "überraschend" gut. Selbst für eine Lageregelstrecke eines senkrecht startenden Flugmodells mit doppeltem integralen Verhalten ergab sich in der Tat ein sehr befriedigendes Führungsverhalten. Störungen hingegen wurden nur sehr träge ausgeregelt. Im Vergleich mit anderen Reglern der gleichen Ordnung war das Verhalten aber ausgezeichnet.

Eine sehr direkte Methode, diskrete Regler PID-Regler aus kontinuierlichen zu erhalten, besteht in der Approximation der Differentialgleichung eines kontinuierlichen PID-Reglers durch eine Differenzengleichung [F4, 11]. Es sind verschiedene Methoden angegeben worden, die sich in erster Linie nur durch die Art der Approximation - Rechtecknäherung, Trapezformel, Tustinformel etc. - unterscheiden und die oftmals nur geringfügige Unterschiede in den Koeffizienten des diskreten Reglers besitzen. Auf diese Weise entworfene Regler arbeiten quasikontinuierlich, sodaß der Bereich der möglichen Abtastzeiten nach größeren Zeiten hin eingeschränkt ist.

Als Einstellhilfen für kontinuierliche oder quasikontinuierliche PI- oder PID-Regler wurden viele Regeln angegeben, die sich leicht in entsprechende Programme fassen lassen :

- Einstellregeln nach Takahashi [T3],
- Ziegler/Nichols - Regeln [Z2],
- Optimierung mittels einer Vergleichsregelstrecke nach den ISE-, IAE- oder ITAE-Kriterien [M4],
- Einstellregeln nach dem Betragsoptimum [F6],
- Einstellregeln nach Cohen und Coon [C3].

Andere einfache Reglerentwürfe gehen von einer Regelstrecke mit einem PT_1T_2 -Verhalten aus. Versucht man die Dynamik der Regelstrecke zu kompensieren - mit Ausnahme der Totzeit natürlich - so gelangt man zu einem Vorschlag von Dahlin. Oft liegt eine Nullstelle der Strecke nahe des Einheitskreises und sollte, um störende Schwingungen zu vermeiden, nicht kompensiert werden, woraus eine Modifikation des Algorithmus folgt [B6]. Verlangt man, daß für die ersten N Abtastschritte eine konstante Stellgröße u_0 vom Regler ausgegeben wird, so erhält man einen Stellgrößen-Vorgabe- (SV-) Algorithmus nach Leonhard [L1]. Unter Einbezug eines Haltegliedes durch eine Näherung in Form einer Totzeit und einer Prädiktion des Streckenverhaltens können ein Proportional-Prädiktor-Regler und mit einem zusätzlichen I-Anteil ein Proportional-Integral-Prädiktor-Regler entworfen werden [M5].

Bereits in den fünfziger Jahren wurde von Bergen und Ragazzini ein Regelkonzept vorgeschlagen, welches das Verhalten der Regelstrecke zu kompensieren trachtet und eine Vorgabe des Führungsverhaltens erlaubt [B8]. In der Folgezeit ist dieser Vorschlag immer wieder aufgegriffen und für eine große Zahl von Reglerentwürfen mit Erfolg eingesetzt worden [B5]. Bei diskreten Reglern bereitet die Wahl der Abtastzeit manchesmal Probleme. Nach unten hin ist sie einerseits durch die Rechengeschwindigkeit des Prozessors und - meist in stärkerem Maße - durch die auftretenden Stellgrößen begrenzt. Letzteres gilt ganz besonders für die noch zu besprechenden Regler mit endlicher Einstellzeit. Kritisch ist die untere Grenze jedoch meist nicht, da man eher an einer großen Abtastzeit interessiert ist. Ein ganz wesentlicher Anhaltspunkt für die obere Grenze der Abtastzeit ergibt sich aus einer Betrachtung des Störverhaltens. Im ungünstigsten Fall wird eine eingetretene Störung erst nach Ablauf eines Abtastschrittes zu einer Reaktion des Reglers führen. Die in dieser Zeit mögliche Änderung des Ausgangs-

signales der Regelstrecke bei einer signifikanten Störung und die erforderliche Regelgüte liefern eine obere Schranke für die Abtastzeit [S3].

Zum Entwurf wird das Führungsverhalten K_w des geschlossenen Kreises mit

$$K_w(z) = \frac{R(z)G(z)}{1 + R(z)G(z)}$$

vorgegeben. Aus Bedingungen für die Kausalität der Reglerübertragungsfunktion, ihrer Realisierbarkeit und einer verschwindenden, bleibenden Regelabweichung ergeben sich Bestimmungsgleichungen für den Regler. Falls die Regelstrecke Pole und/oder Nullstellen außerhalb des Einheitskreises besitzt, so sind diese durch eine Aufspaltung der Zähler- und Nennerpolynome der Regelstrecke gesondert zu behandeln, da Kompensationen hier nicht zulässig sind [B5, U4]. Man definiert

$$G(z) = \frac{B^+(z)B^-(z)}{A^+(z)A^-(z)} z^{-d},$$

wobei die Teilpolynome A^- und B^- die Wurzeln außerhalb und auf dem Einheitskreises enthalten. Es ergibt sich dann für die Führungsübertragungsfunktion

$$K_w(z) = \frac{B_k(z)P(z)B^-(z)z^{-d}}{N(z)}.$$

Das charakteristische Polynom $N(z)$ des geschlossenen Regelkreises kann beliebig vorgegeben werden. Eine interessante Möglichkeit sind diskretisierte Normpolynome [U3, U6] wie sie auch bei kontinuierlichen Entwürfen eingesetzt werden. Das Polynom $B_k(z)$ erlaubt es, einen Einfluß auf die Stellgröße zu

nehmen. Im einfachsten Fall mag man es zu $B_k = 1$ wählen. Eine weitere Bestimmungsgleichung

$$1 - K_w(z) = \frac{A^-(z) (1-z^{-1})^r Q(z)}{N(z)}$$

führt schließlich auf die eigentliche Bestimmungsgleichung der Koeffizienten der Polynome $P(z)$ und $Q(z)$

$$N(z) - A^-(z) (1-z^{-1})^r Q(z) = B^-(z) B_k(z) P(z) z^{-d}$$

und damit zur Reglerübertragungsfunktion

$$R(z) = \frac{A^+(z) B_k(z) P(z)}{B^+(z) Q(z) (1-z^{-1})^r}$$

Wählt man die Führungsübertragungsfunktion als ein endliches Polynom, d.h. das Polynom $N(z) = 1$, so gelangt man zur Klasse der Reglerentwürfe auf endliche Einstellzeit (Dead-Beat-Regler). Meist verknüpft man mit den bereits erwähnten weitere Forderungen. Um ein Schwingen der Regelgröße zwischen den Abtastzeitpunkten zu unterbinden, muß auch die Stellübertragungsfunktion ein endliches Polynom sein. Das Polynom $B_k(z)$ kann dann auch so vorgegeben werden, daß die ersten Abtastschritte nach einer Führungsgrößenänderung festgelegt werden [Z4].

Eine Variation dieser Methode erlaubt einen entkoppelten Entwurf von Führungs- und Störverhalten [A3, A7]. Damit ist es beispielsweise denkbar, als Führungsübertragungsfunktion eine endliche Einstellzeit und für das Störverhalten das eines PID-Reglers vorzuschreiben. Auf diese Weise gelangt man zu Modellfolge-regelkonzepten [A7].

6. 2. Regelung mit Zustandsraummethoden

Neben der Beschreibung durch Übertragungsfunktionen ist die Zustandsraumdarstellung die wichtigste und leistungsfähigste Form zur Behandlung dynamischer Systeme. Vor allem die Erweiterung auf Mehrgrößensysteme kann hier sehr leicht erfolgen. Zustandsraumdarstellungen sind daher ein ganz wesentliches Werkzeug der Regelungstechnik geworden.

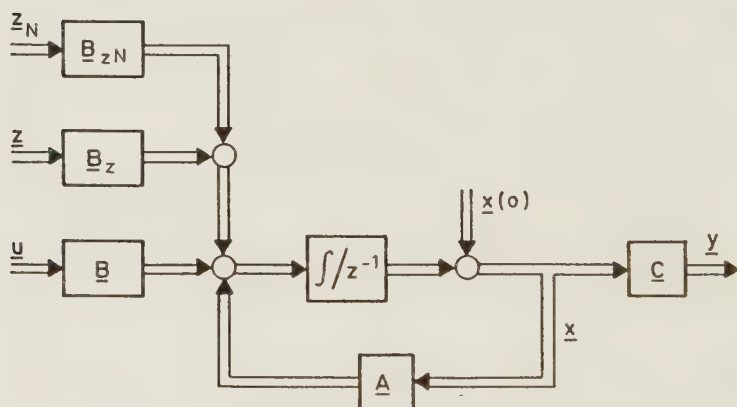


Bild 6.2.1 Zustandsraumdarstellung einer Regelstrecke mit meßbaren und nicht meßbaren Störungen

Die Regler- und Beobachtersynthese im Zustandsraum ermöglicht zumindest prinzipiell, die Dynamik des Regelkreises nahezu beliebig vorzugeben. Ausgangspunkt der Betrachtung ist eine kontinuierliche oder diskrete Regelstrecke, wie sie in Bild 6.2.1 dargestellt ist. In der kontinuierlichen Form gilt

$$\dot{\underline{x}}(t) = \underline{A} \underline{x}(t) + \underline{B} \underline{u}(t) + \underline{B}_z \underline{z}(t) + \underline{B}_{zN} \underline{z}_N(t)$$

$$\underline{y}(t) = \underline{C} \underline{x}(t)$$

In Erweiterung der sonst üblichen Darstellung [A2, I1, L2, U4] findet man neben den Matrizen A, B und C zwei weitere Eingangsmatrizen B_z für meßbare und B_{zw} für nicht meßbare Störgrößen. Die entsprechende diskrete Beschreibung lautet

$$\underline{x}(k+1) = \underline{A} \underline{x}(k) + \underline{B} \underline{u}(k) + \underline{B}_z \underline{z}(k) + \underline{B}_{zw} \underline{z}_w(k)$$

$$\underline{y}(k) = \underline{C} \underline{x}(k) \quad .$$

Im folgenden wird zwischen der kontinuierlichen und der diskreten Darstellung nicht mehr weiter unterschieden, da viele Betrachtungen völlig analog verlaufen [A1, U5]. Soweit nicht die Gefahr von Verwechselungen besteht, wird die diskrete Form der besseren Übersichtlichkeit halber verwendet werden. Die meisten Betrachtungen sind auch für den kontinuierlichen Fall gültig. Außerdem werden in Zukunft die Anfangsbedingungen nicht mehr dargestellt, um die Bilder etwas übersichtlicher zu halten. Trotzdem muß man die kontinuierliche und die diskrete Form sauberlich unterscheiden, wenn man beispielsweise Überlegungen zur Lage von Eigenwerten anstellt oder Pole und Nullstellen betrachtet. Auch die Anfangsbedingungen dürfen nicht vergessen werden.

Die Grundaufgabe des Zustandsreglerentwurfes ist es, durch eine passende Rückführung des Streckenzustands x über eine Reglermatrix F_x auf den Streckeneingang

$$\underline{u}(k) = - \underline{F}_x \underline{x}(k)$$

die Dynamik des geschlossenen Regelkreises nach bestimmten Kriterien zu verändern.

Für eine praktische Anwendung reicht diese einzelne Rückführung meist nicht aus. Sie wird vielmehr durch weitere Maßnahmen ergänzt - vgl. Bild 6.2.2.

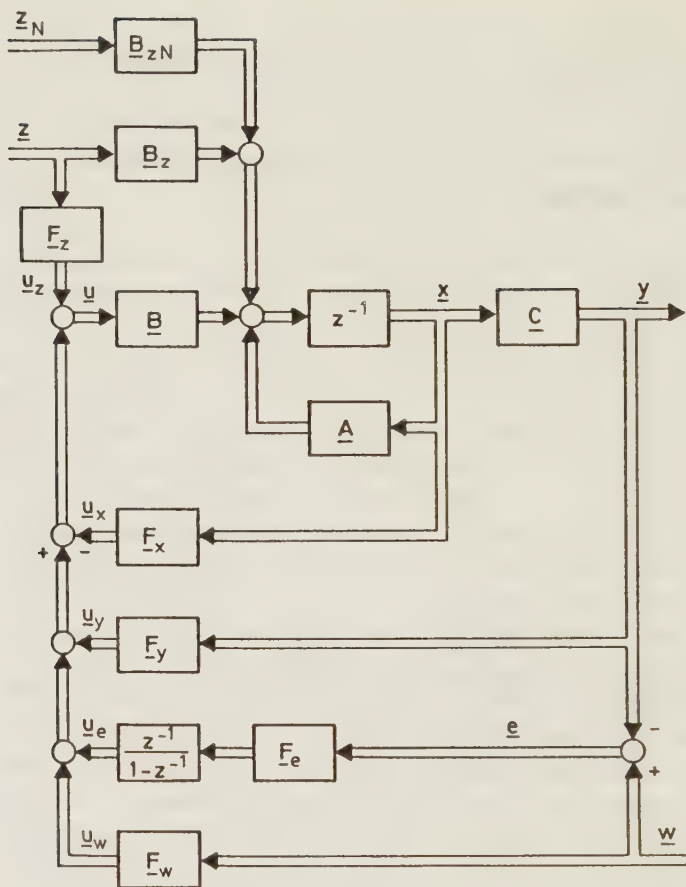


Bild 6.2.2 Struktur einer erweiterten Zustandsraumregelung

Über ein Vorfilter $\underline{F}_w$ wird der Sollwertvektor angeschlossen. Eine Störgrößenaufschaltung kann über die Matrix $\underline{F}_z$ erfolgen. Für einzelne Zustandsgrößen oder auch Linearkombinationen aus ihnen kann man ersatzweise die direkte Rückführung der Ausgangsgrößen der Regelstrecke über die Matrix $\underline{F}_y$ vorsehen. Prinzipiell ist $\underline{F}_y$ aber eine redundante Rückführung, die erst mit der Verwendung reduzierter Beobachter zwingend erforderlich wird. Zuletzt kann die Regelabweichung $\underline{e}$, gebildet aus der Differenz von Sollwert $\underline{w}$ und Istwert $\underline{y}$, über eine Matrix $\underline{F}_e$ und eine anschließende Inte-

gration zur Bildung der Stellgröße herangezogen werden. Hiermit ist es auch bei konstanten Störungen möglich, eine verschwindende Regelabweichung zu erzielen. Man spricht daher auch von einer PI-Zustandsregelung [S5, U5]. Bild 6.2.2 stellt bereits eine für praktische Zwecke notwendige Erweiterung der üblichen Strukturen dar, obgleich sie zunächst noch unterstellt, der Streckenzustand sei vollständig meßbar und eine Störgrößenaufschaltung erfolge nur mit den meßbaren Störungen, also ohne irgendwelche geschätzten Störgrößen. Es gelten folgende Differenzengleichungen

für die Regelstrecke

$$\underline{x}(k+1) = \underline{A} \underline{x}(k) + \underline{B} \underline{u}(k) + \underline{B}_z \underline{z}(k) + \underline{B}_{zN} \underline{z}_N(k)$$

$$\underline{y}(k) = \underline{C} \underline{x}(k)$$

und für den Regler

$$\underline{p}(k+1) = \underline{p}(k) + \underline{e}(k)$$

$$\underline{u}(k) = - \underline{F}_x \underline{x}(k) + \underline{F}_y \underline{y}(k) + \underline{F}_e \underline{e}(k)$$

$$+ \underline{F}_w \underline{w}(k) + \underline{F}_z \underline{z}(k) .$$

Normalerweise ist der Streckenzustand $\underline{x}$ nicht als Ganzes zugänglich, sodaß seine unmittelbare Rückführung über die Matrix $\underline{F}_x$ nicht möglich ist. Diesem Umstand hilft man mit der Einführung eines Beobachters ab, der einen geschätzten Zustandsvektor liefert, welcher ersatzweise zur Regelung dient. Man zeigt leicht, daß sich die gewünschten Eigenwerte des Systems hierbei nicht verändern, sondern daß nur die frei wählbaren Eigenwerte des Beobachters hinzukommen [A2, U5]. Vielfach wird für den Beobachter gefordert, daß er unmittelbar den geschätzten Streckenzustand liefert, was zumindest für die Aufzeichnung der auftretenden Signale einen gewissen Vorteil mit sich bringt. Grundsätzlich kann aber auch jede reguläre Trans-

formation des Streckenzustandes als Beobachterzustandsvektor $\underline{v}$ verwendet werden, wie es bereits in dem grundlegenden Aufsatz von Luenberger dargestellt wurde [L4]

$$\underline{v}(k) = \underline{T} \underline{x}(k) .$$

Die Transformationsmatrix $\underline{T}$ ist bei der Realisierung des Beobachters zu berücksichtigen, muß aber nicht zusätzlich zu jedem Abtastzeitpunkt gerechnet werden, da sie in andere Matrizen hineinmultipliziert werden kann.

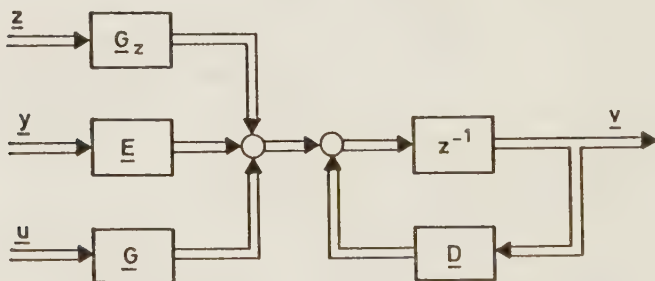


Bild 6.2.3 Luenberger Beobachter, hier ohne Matrix zur Rücktransformation dargestellt

Bild 6.2.3 zeigt einen Beobachter mit dem Beobachterzustandsvektor $\underline{v}$. Ergänzt wurde er bereits durch eine Aufschaltung der meßbaren Störgrößen, die nicht in jedem Falle nötig ist. Die erforderliche Matrix zur Rücktransformation ist aber nicht mit dargestellt. Im allgemeinen Fall kann der Beobachter auch noch Störungen approximieren, sodaß die Rücktransformation neben dem Streckenzustand einen weiteren Vektor zur Störgrößenaufschaltung liefert. Außerdem ist es möglich, einen reduzierten Beobachter zu verwenden, sodaß der Streckenzustand erst durch die gemeinsame Rücktransformation von Beobachterzustandsvektor und Streckenausgang erhalten wird. Für einen reduzierten Beobachter, der zugleich eine Schätzung der nicht meßbaren Störungen liefert und selbst eine Störgrößen-

aufschaltung mit den meßbaren Störungen enthält, gilt dann

$$\underline{v}(k+1) = \underline{D} \underline{v}(k) + \underline{E} \underline{y}(k) + \underline{G} \underline{u}(k) + \underline{G}_z \underline{z}(k)$$

$$\begin{bmatrix} \underline{x}(k) \\ \underline{z}_M(k) \end{bmatrix} = \begin{bmatrix} \underline{T} \\ \underline{C} \end{bmatrix}^{-1} \begin{bmatrix} \underline{v}(k) \\ \underline{y}(k) \end{bmatrix}.$$

In Bild 6.2.4 wird die letzte Gleichung verdeutlicht. Die Zusammenfassung zweier Vektoren zu einem einzigen und die Zerlegung eines Vektors in zwei Teilvektoren kann prinzipiell durch passende Matrizen in der mathematisch richtigen Weise dargestellt werden [H6]. Diese zusätzlich erforderlichen Matrizen und damit verbundene Summenpunkte und Verzweigungspunkte komplizieren die Darstellung aber so erheblich, daß die weniger gebräuchliche Form von Matrizen mit zwei vektoriellen Ein- oder Ausgängen übersichtlicher erscheint und im folgenden erforderlichenfalls bevorzugt wird.

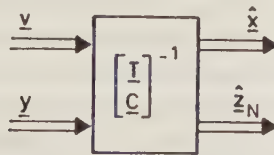


Bild 6.2.4 Rücktransformation von Beobachterzustands- und Streckenausgangsvektor zum Streckenzustands- und nicht meßbaren Störvektor

Eine weitere Komplikation ergibt sich aus der Tatsache, daß zur Störgrößenaufschaltung seitens des Reglerentwurfs nur eine einzelne Matrix $\underline{F}_z$ vorgesehen ist. Wenn eine Störgrößenaufschaltung auch mit geschätzten, nicht meßbaren Störgrößen erfolgen soll, so sind diese während des Reglerentwurfs als meßbar zu betrachten. Erst im realisierten Regler werden dann die beiden Teilvektoren der meßbaren Störung und der geschätzten nicht meßbaren Störung zu einem Gesamt-

vektor zusammengefaßt, der über die Matrix F_z zur Stellgröße beiträgt, wie es in Bild 6.2.5 gezeigt ist.

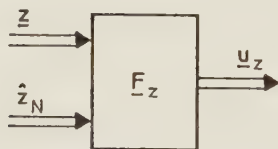


Bild 6.2.5 Erzeugung der Störgrößenaufschaltung

Um mit dem hier vorgestellten Konzept eine Zustandsregelung in möglichst weitgehend automatisierter Form zu entwerfen, ist eine tiefreichende strukturelle Gliederung der auftretenden Vektoren und der jeweiligen Systeme erforderlich, die zur Berechnung des Beobachters und des Reglers benötigt werden. Zuerst wird die Struktur des gesamten Regelsystems definiert. Hierzu gehören die Anzahl von Regelgrößen, die nicht notwendigerweise gleich der Anzahl der Ausgangsgrößen sein muß, und die Anzahl der Störgrößen unabhängig davon, ob sie meßbar sind oder von einem Beobachter geschätzt werden. Weiterhin ist festzulegen, ob ein PI-Regler entworfen werden soll. Wenn dies der Fall ist, dann ist die Dimension des Fehlervektors e gleich der Anzahl der Regelgrößen und maximal gleich der Dimension des Ausgangsvektors der Regelstrecke. Schließlich muß definiert werden, ob ein Beobachter im Regelkreis vorhanden sein soll oder ob gemessene Zustandsgrößen vorliegen. Falls ein Beobachter eingesetzt wird, wie es wohl meistens der Fall sein wird, so ist seine Dimension festzulegen. Hierzu kann man zunächst einmal zwischen einem vollständigen und einem reduzierten Beobachter wählen. Ein vollständiger Beobachter besitzt für die Schätzung des Streckenzustandes einen Anteil mit der gleichen Dimension wie die Regelstrecke selbst. Ein reduzierter Beobachter minimaler Ordnung muß zur Zustandsschätzung gerade eine Ordnung besitzen, die der Differenz aus der Dimension der Strecke und der Dimension des Streckenausgangs entspricht. Falls die Anzahl

von Regelgrößen und die Anzahl der Streckenausgänge verschieden sind, tritt hier eine sehr typische Situation auf. Es werden dann nämlich ein oder auch mehrere Ausgänge zwar zur Zustandsschätzung durch den Beobachter verwendet, nicht aber für die Regelung benutzt. Unabhängig von der Art des Beobachters - vollständig oder reduziert - kann zusätzlich eine Schätzung von Störgrößen erfolgen. Dadurch erhöht sich die Dimension des Beobachters entsprechend der Anzahl der mitgeschätzten Störgrößen und überschreitet dann die Dimension der Regelstrecke bzw. die bereits erwähnte Differenz der Dimensionen des Zustands- und des Ausgangsvektors.

Es treten also folgende Vektordimensionen auf als Anzahl von

- Zustandsgrößen der Regelstrecke
- Stellgrößen
- Ausgangsgrößen der Regelstrecke
- Regelgrößen
- Regelabweichungen
- gemessenen Störgrößen
- Zustandsgrößen des Beobachters
- geschätzten Störgrößen
- geschätzten Zustandsgrößen .

Diese Dimensionen können nicht willkürlich festgelegt werden, sondern stehen teilweise in festen Beziehungen zueinander. So darf weder die Anzahl der Stellgrößen noch die der Ausgangsgrößen die Dimension der Regelstrecke überschreiten - sie wären sonst redundant. Falls eine PI-Regelung gewünscht wird, so ist für den Reglerentwurf, nicht aber für den Beobachterentwurf eine Erweiterung der Regelstrecke mit der Dimension der Regelgrößen nötig. Die Regelgrößen sind ausgewählte oder auch alle Ausgangsgrößen des Systems. Dies bedeutet insbesondere, daß es durchaus sinnvoll ist, Ausgangsgrößen in einem System zu haben, die keine Regelgrößen sind, wohl aber für die Zustandsschätzung

eingesetzt werden. In einer Störgrößenaufschaltung tritt als Dimension die Summe der Anzahl der gemessenen und der geschätzten Störgrößen auf. Und die Dimension des Beobachters setzt sich aus der Dimension der eventuell mitgeschätzten Störgrößen und der Anzahl der geschätzten Zustandsgrößen zusammen. Letzere sind hier immer auch in transformierter Form zugelassen. Die Anzahl der geschätzten Zustände der Regelstrecke ist bei einem vollständigen Beobachter gleich der Dimension der Strecke ohne die Erweiterung für die PI-Regelung. Bei einem reduzierten Beobachter minimaler Ordnung vermindert sich die Dimension des Beobachters um die Anzahl der gemessenen Ausgangsgrößen des Systems.

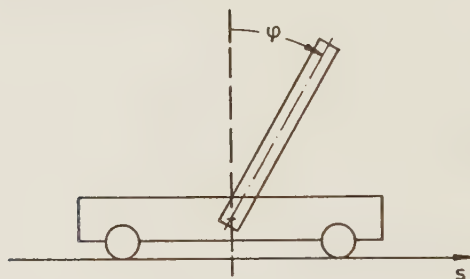


Bild 6.2.6 balancierter Stab

Bild 6.2.6 zeigt ein Beispiel für ein System, welches zwei Ausgangsgrößen besitzt, von denen aber nur eine geregelt werden kann. Es handelt sich um eine Modellbeschreibung eines balancierten Stabes in einer Richtung. Verwendet man als Zustandsvektor des Systems

$$\underline{x}(t) = [\dot{\varphi} \quad \varphi \quad \dot{s} \quad s]^T,$$

wobei φ den Winkel des Stabes gegen die Vertikale und s den Weg des Wagens darstellen, so erhält man für ein Labormodell die Zustandsraumdarstellung [K2, W6]

$$\dot{\underline{x}}(t) = \begin{bmatrix} 0 & 0 & 2,1798 & 0 \\ 0 & 0 & 0,2873 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \underline{x}(t) + \begin{bmatrix} 0,5105 \\ 1,8712 \\ 0 \\ 0 \end{bmatrix} \underline{u}(t)$$

$$\underline{y}(t) = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \underline{x}(t)$$

Man erkennt an der Matrix $\underline{C}$ sofort, daß als Ausgangsgrößen die Position des Wagens und der Winkel des Stabes gemessen werden. Während aber die Wagenposition frei vorgegeben werden kann und daher eine Regelgröße ist, kann man für den Winkel nur den Wert Null als stationären Wert zulassen. Es ist physikalisch nicht möglich, zugleich die Position des Wagens konstant zu halten, während der am Stab gemessene Winkel von Null verschieden ist. Der Stab würde zwangsläufig umfallen. Ein Sollwert darf also nur für die Wagenposition vorgesehen werden. Der gemessene Winkel dient dann nur noch im Beobachter zur Zustandsschätzung. Für den Reglerentwurf verwendet man dann gemeinsam mit einer PI-Zustandsregelung eine Matrix $\underline{C}$, die nur eine Zeile enthält

$$\underline{C} = [0 \ 0 \ 0 \ 1] .$$

Für den Beobachterentwurf setzt man aber die ursprüngliche Matrix $\underline{C}$ ein. Da der Zustandsregler das System stabilisiert, erfolgt natürlich auch eine Regelung des Stabwinkels, allerdings nur auf den Wert Null. Bild 6.2.7 zeigt den Verlauf der Wagenposition für eine sprungförmige Sollwertänderung unter Verwendung eines vollständigen Beobachters. Die Regelstrecke wurde mit einem Analogrechner simuliert, der Regler und der Beobachter sind ein Mikrorechnerprogramm [K2].

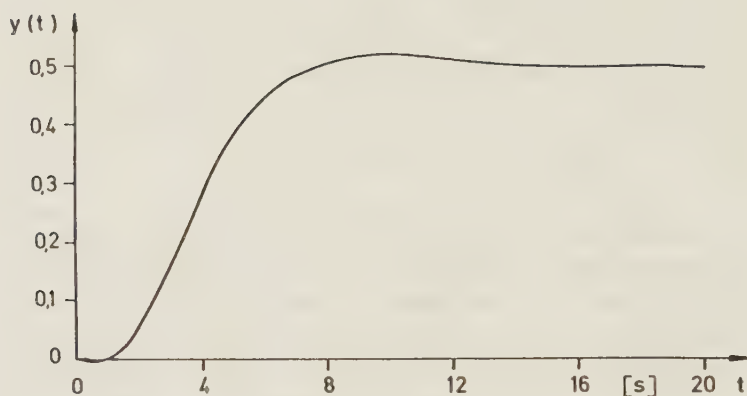


Bild 6.2.7 Regelverhalten des balancierten Stabes mit einem PI-Zustandsregler und Beobachter

Ausgangspunkt der Zustandsregelung bildet eine Gliederung der erforderlichen Gesamtstruktur. In einem Konfigurationsprogramm **KORB** werden alle notwendigen Angaben interaktiv erfragt und hieraus eine Konfigurationsdatei erstellt. Neben den oben angegebenen Dimensionen enthält sie die Namen der Dateien, welche zur Speicherung der verschiedenen Teile der Regelung nötig sind. Als erstes zählt hierzu die ursprüngliche Zustandsraumdarstellung der Regelstrecke. Abhängig von den jeweils gewählten optionalen Eigenarten der Regelung wird ein Reglerentwurfssystem und ein Beobachterentwurfssystem dort verzeichnet sein. Die einzelnen Matrizen des Reglers werden auf einzelnen Dateien getrennt abgelegt, da sie zumeist nicht durch ein einziges Entwurfsprogramm entstehen, sondern nacheinander aufeinander aufbauen oder sogar voneinander unabhängig berechnet werden können. Die Matrizen des Beobachters, sofern ein solcher vorhanden ist, werden hingegen gemeinsam und einschließlich der Transformationsmatrix in einer Datei abgelegt, weil sie bei der Berechnung immer gleichzeitig anfallen. Die nachfolgenden Regler- und Beobachterentwürfe greifen stets auf die Konfigurationsdateien zurück und automatisieren die

Datenhaltung damit ganz wesentlich. Außerdem wird einem Benutzer nicht mehr zugemutet, die Vielzahl der Matrizen passend zusammenzustellen und die Teilvektoren richtig zusammenzusetzen.

6. 2. 1. Zustandsregler

Dem Entwurf von Zustandsreglern dienen eine ganze Reihe von verketteten Einzelprogrammen :

- KORB - Konfigurierung der Regelung
- STEUER - Steuerbarkeitsanalyse des Systems
- EINER - Zustandsvektorrückführung vom Rang 1
- STAGE - stationär genaue Zustandsregler
- VOR - Reglerentwurf durch Vorgabe einer charakteristischen Polynommatrix
- RICCATI - quadratisch optimaler Zustandsregler
- FILTER - Vorfilterentwurf und Störgrößenaufschaltung.

Alle diese Programme sind von einem übergeordneten Menue REIZ aus zugänglich [W3]. Dieses ist auch vorbereitet, neue Reglerentwürfe zu implementieren und stellt damit für Ergänzungen eine wichtige Schnittstelle dar. Durch das Zusammenwirken mit der Konfigurationsdatei muß in keinem Reglerentwurf mehr die nötige Regelstruktur ermittelt werden. Sie ist nach Bedarf festgelegt und damit sofort zugänglich.

Für alle Reglerentwürfe ist die wichtigste Voraussetzung die Steuerbarkeit des Reglerentwurfssystems [A2, U4 et al]. Sie ist gegeben, wenn die Hypermatrix

$$(A, B) = \begin{bmatrix} B & A*B & A^2*B & \dots & A^{n-1}*B \end{bmatrix}$$

n linear unabhängige Spalten besitzt, also regulär ist.

Die Zahl n ist dabei die Ordnung des Systems. Die Reihenfolge der Spalten ist zur Überprüfung der Steuerbarkeit zunächst einmal unwesentlich.

Es kann aber auch sein, daß man sich für die Steuerbarkeit des Systems etwas detaillierter interessiert. Betrachtet man die Spaltenvektoren der Eingangsmatrix jeweils für sich und bildet die Vektorkette

$$b_1 \quad A*b_1 \quad A^2*b_1 \quad \dots \quad A^{n-1}*b_1,$$

sodaß n_1 die maximale Anzahl linear unabhängiger Vektoren jeder Vektorkette ist, so erhält man mit jedem n_1 den Steuerbarkeitsindex der i -ten Eingangsgröße. Auch zur Bildung einer quadratischen Matrix aus den möglichen Spaltenvektoren sind verschiedene Lösungen möglich, sei es, daß man aus der Steuerbarkeitsmatrix (A, B) von links beginnend linear unabhängige Vektoren auswählt, oder daß man zunächst nur solche Spalten auswählt, die b_1 enthalten, und dann zu den übrigen fortschreitet.

Im Programm EINER wird als ein erstes Reglerentwurfsverfahren eine Zustandsrückführung F_x gesucht, die sich als dyadisches Produkt zweier Vektoren

$$F_x = g * k^T$$

ergibt, die also vom Rang 1 ist.

Der Vektor g kann hierbei weitgehend frei festgelegt werden. Zur Berechnung des Vektors k wird ein Polvorgabeverfahren für das eingangstransformierte System genutzt. Voraussetzung ist es, daß das System zyklisch ist, d. h. es darf keine parallelen, nicht verkoppelten Teilsysteme besitzen. Zur Überprüfung berechnet man für jeden Eigenwert s_i den Rangdefekt der Matrix $A - s_i * I_n$, der gerade 1 sein muß [A2, G5]. Ist das System nicht zyklisch, so müssen die Eigenwerte zunächst durch eine

frei wählbare Rückführung $\underline{F}_{x1}$ separiert werden. Für das nunmehr oder von Anfang an zyklische System wird der Vektor $\underline{k}$ bestimmt zu

$$\underline{k}^T = (\underline{p}^T - \underline{q}^T) * \underline{W}^{-1}$$

mit $\underline{p}$ und $\underline{q}$ den Koeffizientenvektoren der charakteristischen Gleichungen des geschlossenen und des offenen System und der Matrix $\underline{W}$ aus

$$\underline{W} = \underline{Q}_{ab} * \begin{bmatrix} 1 & 0 & . & . & 0 \\ q_{n-1} & 1 & & & . \\ . & . & . & & . \\ . & . & & . & 0 \\ q_1 & q_2 & . & q_{n-1} & 1 \end{bmatrix}$$

mit

$$\underline{Q}_{ab} = \begin{bmatrix} \underline{A}^{n-1} * \underline{B} * \underline{g} & \underline{A}^{n-2} * \underline{B} * \underline{g} & . & \underline{B} * \underline{g} \end{bmatrix} .$$

Als Rückführmatrix ergibt sich dann

$$\underline{F}_x = \underline{g} * (\underline{p}^T - \underline{q}^T) * \underline{W}^{-1}$$

oder

$$\underline{F}_x = \underline{g} * (\underline{p}^T - \underline{q}^T) * \underline{W}^{-1} + \underline{F}_{x1} ,$$

falls das System zunächst zyklisch gemacht werden mußte.

Mit einer solchen Regelung können zwar die Pole des geschlossenen Regelkreises an eine beliebige Stelle verschoben werden, doch ist für eine Führungsgrößenregelung immer noch ein Vorfilter $\underline{F}_w$ notwendig, damit die stationären Endwerte von Führungsgrößen und Regelgrößen übereinstimmen. Versucht man diese Forderung unmittelbar in den Reglerentwurf einzubringen, indem man das Vorfilter als Einheitsmatrix definiert, so gelangt man zu stationär genauen Zustandsreglern [Pl]. Voraussetzen muß man hierzu, daß die Dimensionen des Eingangs- und des Ausgangsvektors übereinstimmen. Im Programm STAGE wird hiervon und einem Polvorgabeverfahren wie zuvor beschrieben Gebrauch

gemacht.

Unter Verwendung der eindeutigen Moore-Penrose-Pseudoinversen $\underline{B}^+$ der Eingangsmatrix $\underline{B}$ des Systems ergibt sich die Rückführmatrix $\underline{F}_x$

$$\underline{F}_x = \underline{C} + \underline{B}^* \underline{A} + \underline{V}^* (\underline{I}_n - \underline{B}^* \underline{B}^+) \underline{A},$$

worin $\underline{V}$ eine noch frei wählbare Matrix ist, die dem geschlossenen Regelkreis die gewünschte Dynamik verleihen soll. Sie ist zugleich eine Ausgangsvektorrückführung eines fiktiven Systems

$$\underline{x}' = \underline{A}' \underline{x}' + \underline{B} \underline{u}'$$

$$\underline{y}' = \underline{C}' \underline{x}'$$

$$\underline{u}' = - \underline{V} \underline{y}'$$

mit

$$\underline{C}' = (\underline{I}_n - \underline{B}^* \underline{B}^+) \underline{A}$$

und

$$\underline{A}' = (\underline{I}_n - \underline{B}^* \underline{B}^+) \underline{A} - \underline{B}^* \underline{C}.$$

Berechnet man die Rückführmatrix $\underline{V}$ vom Rang 1

$$\underline{V} = \underline{g} * \underline{k}^T$$

als ein dyadisches Produkt, so kann nahezu das gleiche Verfahren wie oben beschrieben zur Polfestlegung verwendet werden. Statt mit dem ursprünglichen System werden alle Berechnungen mit dem fiktiven System ausgeführt, und die Rückführmatrix wird zum Schluß aus den jeweiligen Teilen zusammengefaßt.

Durch das Programm VOR wird ein Zustandsregler unter Verwendung einer Steuerbarkeitsmatrix und Vorgabe einer charakteristischen Polynommatrix berechnet [A2]. Zunächst wird das System durch eine Eingangstransformation in r Teilsysteme so zerlegt, daß jede

der transformierten Eingangsgrößen u_i ein Teilsystem der Dimension n_i steuert. Die Größe der Teilsysteme erhält man aus den Steuerbarkeitsindizes bei der Bildung der Steuerbarkeitsmatrix $\underline{R}$ zu

$$\underline{R} = \begin{bmatrix} \underline{b}_1 & \underline{A}^* \underline{b}_1 & \dots & \underline{A}^{n_1-1} \underline{b}_1 & \underline{b}_2 & \dots & \underline{A}^{n_r-1} \underline{b}_r \end{bmatrix}.$$

Nach Inversion dieser Matrix und zeilenweiser Aufteilung entsprechend der Steuerbarkeitsindizes in $n_i \times n_i$ -Teilmatrizen ergibt sich die Eingangstransformationsmatrix $\underline{M}_{ab}$

$$\underline{M}_{ab} = \begin{bmatrix} \underline{e}_1^T * \underline{A}^{n_1-1} \\ . \\ . \\ \underline{e}_r^T * \underline{A}^{n_r-1} \end{bmatrix} * \underline{B}.$$

Die Zustandsrückführung $\underline{F}_x$ berechnet sich dann aus

$$\underline{F}_x = \underline{M}_{ab} * \begin{bmatrix} \underline{g}_1^T * \underline{K}_1 \\ . \\ . \\ \underline{g}_r^T * \underline{K}_r \end{bmatrix}$$

mit

$$\underline{K}_i = \begin{bmatrix} \underline{e}_i^T \\ \underline{e}_i^T * \underline{A} \\ . \\ . \\ \underline{e}_r^T * \underline{A}^{n_i} \end{bmatrix},$$

worin die Vektoren $\underline{g}_i$ die Koeffizienten der charakteristischen Teilpolynome der r Teilsysteme enthalten. Diese sind durch die Vorgabe von n_i Polen für jedes dieser Teilsysteme bestimmt.

Ein weiteres Programm RICCATI erlaubt den Entwurf von quadratisch optimalen Zustandsreglern [A2, U5]. Mit den Bewertungsmatrizen $\underline{Q}$ für den Zustandsvektor, $\underline{R}$ für den Stellvektor und gegebenenfalls $\underline{M}$ für eine Kreuzbewertung von Zustand und Stellgröße erhält man

die Rückführmatrix $\underline{F}_x$ im Kontinuierlichen zu

$$\underline{F}_x = - \underline{R}^{-1} * \underline{B}^T * \underline{K}$$

bzw. im Diskreten zu

$$\underline{F}_x = - (\underline{R} + \underline{B}^T * \underline{K} * \underline{B})^{-1} * \underline{B}^T * \underline{K} * \underline{A}$$

als Lösungen der Matrix-Riccati-Gleichungen

$$\underline{K} * \underline{A} + \underline{A}^T * \underline{K} - \underline{K} * \underline{B} * \underline{R}^{-1} * \underline{B}^T * \underline{K} + \underline{Q} = \underline{0}$$

bzw.

$$\underline{K} = \underline{Q} + \underline{A}^T * \underline{K} * \underline{A}' - \underline{A}^T * \underline{K} * \underline{B} * (\underline{R} + \underline{B}^T * \underline{K} * \underline{B})^{-1} * \underline{B}^T * \underline{K} * \underline{A} .$$

Die Kreuzbewertungsmatrix $\underline{M}$ läßt sich für diskrete Systeme mit in den Entwurf übernehmen, wenn man folgende Substitutionen vornimmt

$$\underline{A}' = \underline{A} - \underline{B} * \underline{R}^{-1} * \underline{M}$$

$$\underline{Q}' = \underline{Q} - \underline{M}^T * \underline{R}^{-1} * \underline{M} .$$

Einfach zu berechnen ist die stationäre Lösung der kontinuierlichen Matrix-Riccati-Differentialgleichung durch rückwärtige Integration oder rekursiv aus der diskreten Matrix-Riccati-Differenzengleichung.

Ein letztes Entwurfsprogramm **FILTER** dieses Abschnitts dient der Berechnung von Vorfiltern zur Aufschaltung des Führungsgrößenvektors und der Störgrößenaufschaltung, falls durch die Konfigurierung solche erforderlich wurden. Vorausgesetzt wird hierbei, daß bereits ein Regler $\underline{F}_x$ entworfen wurde. Die beiden Aufschaltmatrizen sind dann leicht nach

$$\underline{F}_w = - (\underline{C} * (\underline{S} - \underline{B} * \underline{F}_x)^{-1} * \underline{B})^{-1}$$

$$\underline{F}_z = \underline{F}_w * \underline{C} * (\underline{S} - \underline{B} * \underline{F}_x)^{-1} * \underline{B}_z$$

zu berechnen, wobei für kontinuierliche Systeme $\underline{S} = \underline{A}$ und für diskrete Systeme $\underline{S} = \underline{A} - \underline{I}_n$ zu setzen ist.

6. 2. 2. Beobachterentwurf

Vorangehend wurde bereits erwähnt, daß der Streckenzustandsvektor $\underline{x}$ zum Zwecke der Rückführung über eine Matrix $\underline{F}_x$ in aller Regel nicht direkt zur Verfügung steht, sondern daß er vermittels eines Beobachters geschätzt werden muß. Das Beobachterkonzept geht auf Luenberger zurück [L4, L5]. Ein Beobachter ist ein dynamisches System, welches beginnend mit einem beliebigen Anfangszustand den Zustand der Regelstrecke asymptotisch approximiert, sofern auf das beobachtete System keine anderen Störungen einwirken als solche, die sich durch Anfangswertstörungen ausdrücken lassen.

Für eine Regelstrecke mit der Darstellung

$$\underline{x}(k+1) = \underline{A} \underline{x}(k) + \underline{B} \underline{u}(k)$$

$$\underline{y}(k) = \underline{C} \underline{x}(k)$$

lautet die Differenzengleichung des Beobachters in allgemeiner Form

$$\underline{v}(k+1) = \underline{D} \underline{v}(k) + \underline{E} \underline{y}(k) + \underline{G} \underline{u}(k) ,$$

worin $\underline{v}$ mittels einer Transformation $\underline{T}$ aus dem Zustand der Regelstrecke hervorgeht. Die Regelstrecke kann gegebenenfalls um ein mitgeschätztes Störmodell erweitert worden sein. An den Entwurf eines solchen allgemeinen Beobachters sind verschiedene Bedingungen geknüpft [H6, L4, L5]. Zunächst einmal muß die Regelstrecke beobachtbar sein, welches anhand des Ranges der

Beobachtbarkeitsmatrix $(\underline{C}, \underline{A})$

$$(\underline{C}, \underline{A}) = \begin{bmatrix} \underline{C} \\ \underline{C} * \underline{A} \\ . \\ . \\ \underline{C} * \underline{A}^{n-1} \end{bmatrix}$$

erkennbar ist. Der Rang dieser Matrix muß gleich dem Rang des Systems sein

$$\text{rang}(\underline{C}, \underline{A}) = n.$$

Die aufgeführten Matrizen müssen den Gleichungen

$$\underline{D} * \underline{T} = \underline{T} * \underline{A} - \underline{E} * \underline{C}$$

$$\underline{G} = \underline{T} * \underline{B}$$

genügen.

Diese Gleichungen werden auch zugleich zum Beobachterentwurf herangezogen. Eine weitere Bedingung ist die asymptotische Stabilität des Beobachters selbst, was bedeutet, daß die Eigenwerte der Beobachtermatrix $\underline{D}$ für den diskreten Fall innerhalb des Einheitskreises und für den kontinuierlichen Fall in der linken s-Halbebene liegen. Nur für diesen Fall geht der Beobachtungsfehler als Differenz zwischen wirklichem und rekonstruiertem Zustandsvektor exponentiell gegen Null. Außerdem muß die Matrix $\underline{T}$ gegebenenfalls zusammen mit der Ausgangsmatrix $\underline{C}$ der Regelstrecke bei reduzierten Beobachtern den Rang n besitzen. Bild 6.2.8 zeigt einen vollständigen Beobachter mit der beobachteten Regelstrecke einschließlich der Rücktransformation $\underline{T}^{-1}$.

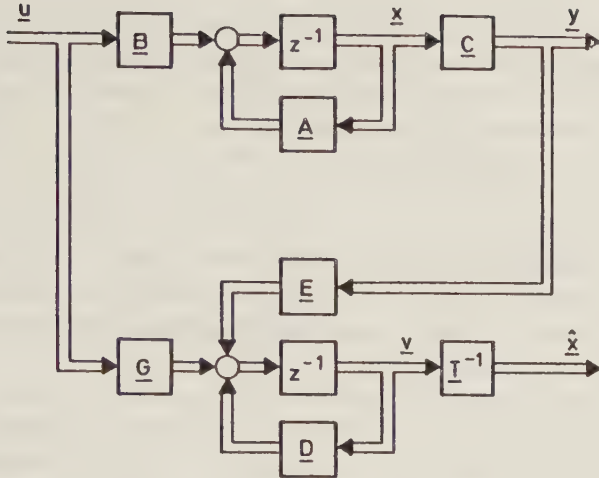


Bild 6.2.8 vollständiger Beobachter mit Regelstrecke

Meßbare Störungen $\underline{z}$, die über die Matrix $\underline{B}_z$ auf die Regelstrecke einwirken, können auch auf den Beobachter als zusätzlicher Eingang aufgeschaltet werden. Das Beobachterentwurfssystem enthält dann eine erweiterte Eingangsmatrix, bestehend aus der ursprünglichen Eingangsmatrix $\underline{B}$ und der Matrix $\underline{B}_z$ [A1]. Nicht meßbare Störungen können durch ein zusätzliches Störmodell als Ergänzung der Regelstrecke mit zum Entwurf herangezogen werden [J2, J3, M6]. Der störende Prozeß wird dazu durch eine Differential- oder eine Differenzengleichung modelliert als

$$\underline{s}(t) = \underline{L} \underline{s}(t) + \delta(t)$$

$$\underline{z}_N(t) = \underline{K} \underline{s}(t)$$

bzw.

$$\underline{s}(k+1) = \underline{L} \underline{s}(k) + \delta(k)$$

$$\underline{z}_N(k) = \underline{K} \underline{s}(k) .$$

Hierin stellen die Deltafunktionen zu beliebigen Zeitpunkten eintretende vektorielle Änderungen des Anfangswertes des Störmodells dar [A1]. Da das Störmodell auch integrale Anteile besitzen darf, sind die approximierten Störungen z nicht auf mittelwertfreie Signale beschränkt.

Ausgangspunkt beim Entwurf von Beobachtern ist wie beim Reglerentwurf eine Konfigurierung der gesamten Regelaufgabe. Der eigentliche Beobachterentwurf läuft dann programmäßig über ein Menueprogramm ab, von dem aus die einzelnen Entwürfe zugänglich sind [K4]. Dieses Menue stellt zugleich die Schnittstelle für spätere Erweiterungen mit ergänzenden Entwurfsverfahren dar. Alle erforderlichen Daten der Beschreibung des Beobachterentwurfssystems werden über die bereits erwähnten Konfigurationsdateien geliefert, sodaß die Struktur der Regelung und damit auch des Beobachters bereits festgelegt ist.

Die einzelnen Teilaufgaben der dem Menue unterlagerten Programme umfassen Methoden für folgende Problemstellungen

- Beobachtbarkeitsanalyse [A2],
- Beobachterentwurf über Kroneckerprodukte [B10],
- Verfahren nach Bongiorno und Youla [B9],
- iteratives Verfahren nach Kreisselmeier [K1],
- Entwurf über Beobachternormalform [A2],
- Beobachtereigenwertfestlegung nach Gopinath [G5].

Eine erste Aufgabe ist die Ermittlung der Beobachtbarkeit des Beobachterentwurfssystems, bzw. die Bestimmung der Beobachtbarkeitsindizes der verschiedenen Ausgangsgrößen. Wenn der Rang der Matrix

$$(\underline{C}, \underline{A}) = \begin{bmatrix} \underline{C} \\ \underline{C} * \underline{A} \\ . \\ . \\ \underline{C} * \underline{A}^{n-1} \end{bmatrix}$$

gleich dem Rang n des Systems ist, dann ist das System beobachtbar. Die Bildung der zu untersuchenden Matrix ist dabei ähnlich wie bei der Analyse der Steuerbarkeit auf verschiedene Weise möglich.

Zum Entwurf eines Beobachters sind geeignete Matrizen $\underline{D}$, $\underline{E}$ und $\underline{T}$ zu finden, sodaß die Matrizen-gleichung

$$\underline{D} * \underline{T} = \underline{T} * \underline{A} - \underline{E} * \underline{C}$$

und einige weitere Bedingungen erfüllt werden. Für diese Beobachterentwurfsgleichung gibt es eine ganze Reihe von Algorithmen. Unterschiedlich ist hierin der Ausgangspunkt, durch den festgelegt wird, welche Matrizen oder andere Eigenschaften - z. B. Beobachter-eigenwerte - vorgegeben werden. Die übrigen Matrizen sind dann jeweils zu berechnen. Grundsätzlich ist die gestellte Aufgabe nicht eindeutig lösbar. Nach Vorgabe einzelner Matrizen kann die Aufgabe entweder eindeutig lösbar, mehrdeutig oder widersprüchlich sein [H6]. Für einen automatisierten Beobachterentwurf wurden solche Lösungswege favorisiert, die zu eindeutig lös-baren Problemstellungen führen.

Ein erstes, sehr direktes Entwurfsverfahren versucht, mit gegebenen Matrizen $\underline{D}$ und $\underline{E}$ die Matrizen-gleichung direkt zu lösen und die Transformationsmatrix $\underline{T}$ zu bestimmen. Eindeutig ist die Lösung genau dann, wenn die Eigenwerte von Beobachter und Regelstrecke verschieden voneinander sind. Dies ist bei bekannten Matrizen $\underline{A}$ und $\underline{D}$ leicht zu überprüfen. Sind n die Ordnung des Systems und r die Ordnung des Beobachters, so wird die Beobachterentwurfsgleichung mit Hilfe des

Kroneckerproduktes in ein inhomogenes, lineares, quadratisches Gleichungssystem rxn -ter Ordnung

$$(\underline{A}^T \otimes \underline{I}_r - \underline{I}_n \otimes \underline{D}) \underline{t} = \underline{f}$$

überführt [B10, N2]. Hierin ist der Vektor $\underline{t}$ durch Aneinanderhängen aller Spaltenvektoren der Matrix $\underline{T}$

$$\underline{t} = \begin{bmatrix} t_{11} & t_{21} & \dots & t_{12} & \dots \end{bmatrix}^T$$

und der Vektor $\underline{f}$

$$\underline{f} = \begin{bmatrix} ec_{11} & ec_{21} & \dots & ec_{12} & \dots \end{bmatrix}^T$$

durch Aneinanderhängen aller Spaltenvektoren des Matrizenproduktes $\underline{E} * \underline{C}$ entstanden.

Grundsätzlich muß nach der Berechnung der Matrix $\underline{T}$ in diesem, wie auch in anderen Verfahren, überprüft werden, ob die Lösung den nötigen Rang besitzt. Für einen vollständigen Beobachter muß

$$\text{rang } \underline{T} = n$$

und für einen reduzierten Beobachter minimaler Ordnung muß

$$\text{rang} \begin{bmatrix} \underline{T} \\ \underline{C} \end{bmatrix} = n$$

gelten. Falls diese Bedingung nicht erfüllt ist, wurde eine ungeeignete Matrix $\underline{E}$ vorausgesetzt. Da es keine unmittelbaren Bedingungen für die Matrix $\underline{E}$ gibt, sollte diese so gewählt werden, daß sie vollen Rang besitzt, damit die Information, die im Ausgangsvektor des Systems enthalten ist, den Beobachter unbeschnitten erreichen kann. Insbesondere sollte keine Zeile der Matrix $\underline{E}$ zu Null gewählt werden, da dies bedeutet, daß der Beobachter ohne die Kenntnis des zugehörigen Streckenausgangs auskommen muß. Im Einzelfall kann

hierin ein Verstoß gegen die Beobachtbarkeitsbedingung liegen.

Ein zweites Beobachterentwurfsverfahren nutzt eine direkte Berechnung der Matrix $\underline{T}$ bei willkürlich gegebener Beobachtermatrix $\underline{D}$ und ebenfalls gegebener Ankopplung an die Regelstrecke über die Matrix $\underline{E}$ nach

$$\underline{T} = -\underline{P}_A^{-1}(\underline{D}) \left(\sum_{j=1}^n a_j \sum_{i=0}^{j-1} \underline{D}^{j-i-1} \underline{E} \underline{C} \underline{A}^i \right)$$

oder

$$\underline{T} = \left(\sum_{j=1}^r d_j \sum_{i=0}^{j-1} \underline{D}^{j-i-1} \underline{E} \underline{C} \underline{A}^i \right) \underline{P}_D^{-1}(\underline{A})$$

aus. Es treten hierin die charakteristischen Gleichungen der Regelstrecke bzw. des Beobachters in normierter Form auf [B9]

$$\underline{P}_A(s) = \det(s\underline{I} - \underline{A}) = \sum_{j=0}^n a_j s^j = 0$$

$$\underline{P}_D(s) = \det(s\underline{I} - \underline{D}) = \sum_{j=0}^r d_j s^j = 0$$

mit

$$a_n = 1 \text{ und } d_r = 1 .$$

Zur Herleitung bedient man sich des Cayley-Hamilton-Theorems, welches auch für den erheblichen Aufwand bei der Berechnung der Summen von Matrizenprodukten verantwortlich ist. Prinzipiell sind zwei verschiedene Berechnungswege möglich. Bei reduzierten Beobachtern ist der zweite aufgezeigte Weg mit geringerem Aufwand verbunden, da die Ordnung r des Beobachters dann kleiner als die Ordnung n des Systems ist. Auch bei dieser Methode muß darauf geachtet werden, daß eine eindeutige Lösung nur dann existiert, wenn Strecken- und Beobach-

tereigenwerte voneinander verschieden sind. Außerdem ist nach erfolgter Berechnung der Matrix $\underline{T}$ der bereits erwähnte Rangtest auszuführen.

Neben direkten Methoden zur Berechnung des Beobachters gibt es verschiedene iterative oder sukzessive Algorithmen. Ein numerisch besonders gutmütiges Verfahren macht von einer Ähnlichkeitstransformation der Beobachterzustandsmatrix $\underline{D}$ Gebrauch

$$\underline{D}_v = \underline{V} \underline{D} \underline{V}^{-1} .$$

Mit den Substitutionen

$$\underline{X} = \underline{V} \underline{T}$$

und

$$\underline{Y} = \underline{V} \underline{E} \underline{C}$$

erhält man formal eine Gleichung, die der ursprünglichen Entwurfsgleichung ähnlich ist,

$$\underline{X} \underline{A} - \underline{D}_v \underline{X} = \underline{Y}$$

aber Freiheiten in der Struktur der Matrix $\underline{D}_v$ läßt [Kl]. Eine mögliche besondere Form der Matrix $\underline{D}_v$ ist die Frobeniusform,

$$\underline{D}_v = \begin{bmatrix} 0 & 0 & \dots & 0 & -d_0 \\ 1 & 0 & & 0 & -d_1 \\ 0 & 1 & & & . \\ . & . & . & 0 & -d_{r-2} \\ 0 & . & . & 1 & -d_{r-1} \end{bmatrix}$$

in der nur die Koeffizienten des charakteristischen Polynoms des Beobachters auftreten. Mit dieser Form kann die letzte Spalte $\underline{x}_r$ der Matrix $\underline{X}$ direkt berechnet werden als Lösung des Gleichungssystems

$$\underline{G}_r \underline{x}_r = \underline{g}_r$$

mit

$$\underline{G}_r = (d_0 \underline{I} + d_1 (\underline{A}^T) + \dots + d_{r-1} (\underline{A}^T)^{r-1} + (\underline{A}^T)^r$$

und

$$\underline{g}_r = \underline{y}_1 + (\underline{A}^T) \underline{y}_2 + \dots + (\underline{A}^T)^{r-1} \underline{y}_r ,$$

worin $\underline{y}_i$ die Spalten der Matrix $\underline{Y}$ sind. Die verbleibenden Spalten der Matrix $\underline{X}$ erhält man sukzessiv nach

$$\underline{x}_{i-1} = \underline{A}^T \underline{x}_i + d_{i-1} \underline{x}_r - \underline{y}_i .$$

Der Beobachterentwurf erfolgt insgesamt in einer sehr unmittelbaren Weise, indem man die Koppelmatrix $\underline{E}$ frei wählt und die Beobachtereigenwerte festlegt. Alle übrigen Berechnungen und der unumgängliche Rangtest können vollständig automatisiert erfolgen.

Bei allen bisher vorgestellten Methoden wurde eine mehr oder weniger willkürlich gewählte Matrix $\underline{E}$ vorausgesetzt. Unabhängig von der Herkunft der Matrix $\underline{D}$ des Beobachters war es das Ziel des Entwurfs, eine Lösung $\underline{T}$ der Entwurfsgleichung zu erhalten, die außerdem nur dann akzeptierbar war, wenn sie der Rangbedingung genügte. Die letzte Bedingung stellt keine besondere Einschränkung dar, weil mit einer ausreichend allgemein festgelegten Ankoppelmatrix $\underline{E}$ die Rangbedingung fast immer erfüllt sein wird. Der Zugang zur Matrix $\underline{D}$ kann genutzt werden, um einen einfachen oder numerisch guten Algorithmus zu erhalten. Die Transformationsmatrix $\underline{T}$ folgt dann durch den Entwurf selbst und ist nicht mehr zu beeinflussen.

Eine andere Vorgehensweise besteht darin, die Matrix $\underline{T}$ a priori als Einheitsmatrix zu definieren. Die Beobachterentwurfsgleichung wird dann zu

$$\underline{D} = \underline{A} - \underline{E} \underline{C} .$$

Obwohl diese Gleichung einfacher aussieht, ist es weitaus schwieriger, sie durch passende Matrizen $\underline{D}$ und

$\underline{E}$ zu lösen. Beobachter, die so entworfen wurden, nennt man Identitätsbeobachter, weil der Zustandsvektor unmittelbar und ohne eine weitere Transformation eine Schätzung des Streckenzustandsvektors ist. Wie man an der Entwurfsgleichung leicht erkennt, ist die Dimension des Beobachters gleich der Dimension der Regelstrecke. Reduzierte Beobachter sind in dieser Form nicht unmittelbar zu erhalten, sodaß man auf vollständige Beobachter beschränkt ist. Der Beobachter darf aber zugleich einen Anteil zur Schätzung von Störgrößen enthalten, da der dimensionsmäßige Anteil des Störgrößenbeobachters auch im zugrundeliegenden Beobachterentwurfssystem enthalten ist. Als Lösungsverfahren wird üblicherweise von einer Regelstreckentransformation in die Beobachternormalform ausgegangen. Die Matrizen $\underline{A}$ und $\underline{C}$ haben dann einen inneren Aufbau, der den Entwurf erheblich unterstützt.

Ein weiteres Beobachterentwurfsprogramm für Identitätsbeobachter verwendet zur Polvorgabe eine Koppelmatrix $\underline{E}$, die als dyadisches Produkt darstellbar ist. Es stellt damit ein Verfahren dar, welches ebenfalls eine Matrix vom Range 1 benutzt und hieraus eine Entwurfsvereinfachung bezieht [G5].

Ähnlich wie man zum Reglerentwurf die Steuerbarkeitsmatrix heranziehen kann, so kann zum Beobachterentwurf die Beobachtbarkeitsmatrix im Entwurf genutzt werden. Beide Entwurfsverfahren sind dual.

Der Entwurf eines Identitätsbeobachters setzte im weiter oben erwähnten Algorithmus als Ziel einen vollständigen Beobachter voraus. Sofern man ein Regelsystem in sogenannten Sensorkoordinaten gegeben hat [$\underline{A2}$], kann auch ein reduzierter Beobachter entworfen werden, der gerade jene Zustangsgrößen schätzt, die nicht im Ausgangsvektor der Regelstrecke enthalten sind [$\underline{C4}$, $\underline{G5}$]. Sensorkoordinaten bedeutet dabei, daß alle Ausgangsgrößen zugleich auch Zustandsgrößen sind. Die Matrix $\underline{C}$ des Systems besitzt also in jeder Zeile genau

einmal den Wert Eins und an allen anderen Positionen Nullen. Erforderlichenfalls kann die Regelstrecke auch auf Sensorkoordinaten transformiert werden, sofern sie nicht in dieser Form vorliegt. In realen, aus mathematisch-physikalischen Modellen gewonnenen Problemstellungen wird die Matrix C oft nur Umrechnungsfaktoren enthalten, die die physikalischen Größen zu elektrischen Meßwerten umskalieren.

Alle Beobachterentwurfsprogramme hinterlegen die erhaltenen Beobachter auf Matrizendateien und tragen in den Konfigurationsdateien die jeweiligen Dateinamen nach, sodaß die zueinander gehörenden Bestandteile einer Zustandsregelung für andere Programme leicht auffindbar sind. Anders als bei den Reglerentwurfsprogrammen werden immer alle Matrizen eines Beobachters D, E, G und T gemeinsam in einer Datei abgelegt, da sie nur gemeinsam verwendbar sind und immer zusammen anfallen.

7. Direkte digitale Regelung

Das Ziel des Rechneinsatzes in der Regelungstechnik ist die direkte digitale Regelung. Für diese Aufgabe ist in jedem Fall ein realisierendes Programm erforderlich. Prinzipiell sollte man hier zwei verschiedene Wege unterscheiden. Zum einen kann für die jeweilige Anwendung ein Programm so angefertigt werden, daß es die gestellte Aufgabe bestmöglich löst. Programme dieser Art zeichnen sich oft durch sehr direkte Vorgehensweisen aus und lassen sich hinsichtlich der Rechenzeit optimieren. Für eine konkrete Aufgabe ist es immer möglich, für fast alle auftretenden Größen Wertebereiche und erforderliche Genauigkeiten anzugeben. Es kann daher lohnenswert erscheinen, zumindest einen Teil der für den "On-line"-Betrieb notwendigen Berechnungen in Form von Festkommaarithmetik durchzuführen, woraus eine oftmals bedeutende Reduzierung der Rechenzeit folgt. Auch die zur Programmflußsteuerung notwendigen Anweisungen können oft verringert werden, da die nötigen Variationsmöglichkeiten des verwendeten Algorithmus auf genau einen einzigen Fall eingeschränkt sind. Spezialprogramme dieser Art besitzen unbestreitbar eine Reihe von Vorzügen, die von einem geschickten Regelungstechniker genutzt werden können. Ihr einziger, aber schwerwiegender Nachteil ist ihre Beschränkung auf eine einzige Problemstellung. Jede Änderung in der Aufgabenstellung an das Regelprogramm wird mit Änderungen des Programmes selbst einhergehen. Und solche Änderungen bedeuten oft genug selbst bei kleineren Modifikationen des angestrebten Zieles tiefgreifende Eingriffe in das Regelprogramm bis hin zu einer Neuprogrammierung ganzer Programmteile. Spezialprogramme sollten daher, wenn irgend möglich, erst dann zur Anwendung kommen, wenn die Aufgabe vollständig gelöst ist und zumindest in einer Simulation zufriedenstellend arbeitet. Auf dem Wege dorthin muß man mit konfigurierbaren, allgemeinverwendbaren Fertigprogrammen arbeiten. Solche Programme lösen eine gestellte Aufgabe während

der Entwicklungsarbeit.

Vorgefertigte Standardprogramme für regelungstechnische Aufgaben stellen eine Softwarebasis dar, die durch den Benutzer auf die jeweilige Anwendung hin eingestellt werden muß. Um leistungsfähig zu sein, sollte ein Programm in diesem Bereich einfach in der Handhabung und umfangreich in seiner Variationsbreite sein. Es sollte also nicht genau ein Regler durch ein Programm realisiert werden, sondern möglichst eine ganze Klasse von strukturell gleichartigen Reglern. Diese können dann anwendungsorientiert mit den erforderlichen Daten versorgt werden. Ein und dasselbe Programm sollte beispielsweise heute als Regler 2. Ordnung, morgen als Regler 4. Ordnung und übermorgen mit einem zusätzlichen Vorfilter arbeiten können, ohne daß Änderungsarbeiten an dem Programm selbst anfallen. Zur Handhabung ist damit eine Interaktion zwischen Programm und Benutzer nötig. Dies darf aber auf keinen Fall heißen, daß der Benutzer die Koeffizienten eines Reglers nebst seiner Ordnung so in einem Menue eingeben sollte, wie sie durch ein Entwurfsprogramm ermittelt wurden. Zweckmäßiger ist es, den Regler über eine Datei vom Entwurfsprogramm zum Regelprogramm zu übertragen. Der Programmbenutzer kann dann mit dem verständlicheren Objekt Regler in der Form eines Dateinamens weitaus angenehmer umgehen als mit einem Satz von Zahlenwerten. Alle vorgefertigten Programme, die an dieser Stelle beschrieben werden, besitzen die Fähigkeit, auf Dateien zuzugreifen und dort die jeweils erforderlichen Daten zu entnehmen.

7. 1. Implementierung von Reglern

Standardbetriebssysteme wie CP/M, MP/M oder MSDOS sind nicht für regelungstechnische Aufgaben entwickelt worden. Dies zeigt sich sofort im Fehlen eines Systemaufrufs, der das rufende oder ein anderes Programm zu äquidistanten Zeitpunkten neu startet. Echtzeitbetriebssysteme der üblichen Prozeßrechner sind in dieser Beziehung wesentlich komfortabler ausgestattet. Prinzipiell stellt aber die Realisierung eines zyklischen Betriebs eines Prozesses kein großes Problem dar.

Ein Verwaltungsprogramm für Echtzeitprozesse RTEL ist in Bild 7.1.1 durch sein Flußdiagramm dargestellt [H1]. Es ist so aufgebaut, daß es nach dem Binden mit den Teilen des Echtzeitprogramms als Hauptprogramm arbeitet. Die regelungstechnische Aufgabe wird hier durch zwei Unterprogramme ausgeführt, die der Initialisierung der erforderlichen Variablen dienen und zum anderen den Algorithmus für jeweils einen Abtastschritt enthalten. Nach einer Selbstinitialisierung (Stackpointer, Uhr, usw.) wird das Unterprogramm zur Initialisierung des Echtzeitprogramms ausgeführt. Hierbei wird vorausgesetzt, daß das Echtzeitprogramm noch keine Daten von seinem Steuerprogramm erhalten hat, und es wird daher auch nicht weiter fortgesetzt, solange nicht von außen, nämlich vom Steuerprogramm aus, ein Start des Regelprogramms veranlaßt wird. Im Betriebssystem MP/M konnten hierfür Systemaufrufe verwendet werden. Nach einem solchen Start erwartet das Regelprogramm einen vollständigen Datensatz, sodaß es seiner Aufgabe gerecht werden kann.

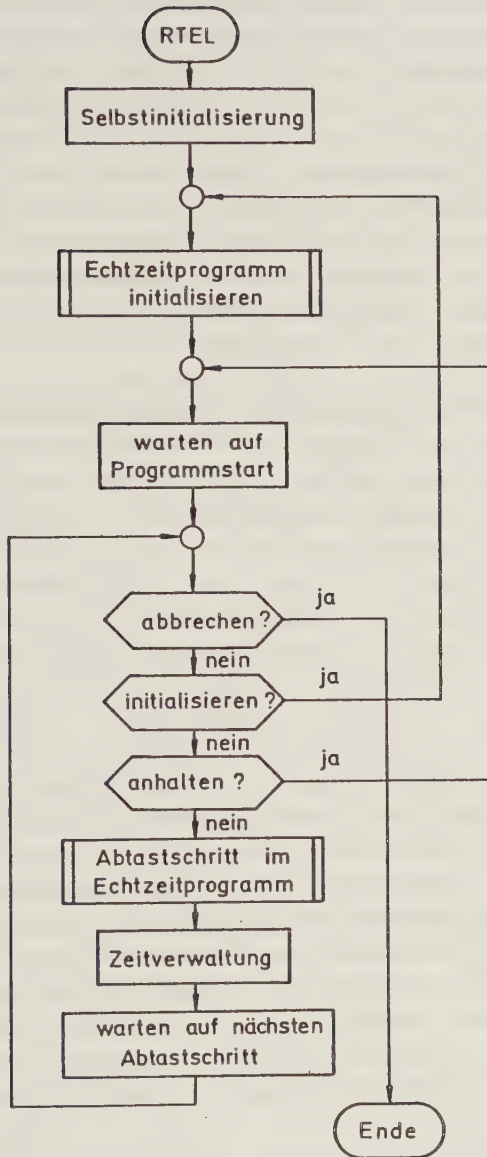


Bild 7.1.1 Flußdiagramm des Verwaltungsprogramms für Echtzeitprogramme

Um zusätzliche Variationsmöglichkeiten zu besitzen, analysiert RTEL zunächst drei Flags, durch die mitgeteilt werden kann, ob es sich terminieren, eine erneute Initialisierung durchführen oder als Normalfall die Berechnungen für einen Abtastschritt ausführen soll. Für die zuletzt genannte Aufgabe muß das eigentliche Regelprogramm als weiteres Unterprogramm aufgerufen werden. Anschließend wird die Zeitverwaltung durchgeführt, die insbesondere auch Synchronfehler erkennt. Nach einem erneuten passiven Warten kann der nächste Abtastschritt bearbeitet werden.

RTEL ist eines der ganz wenigen Assemblerprogramme des gesamten Systems. Und mit etwas mehr als 100 Programmzeilen ist es zugleich auch das größte in Assembler geschriebene Programm. Da es Funktionen enthält, die sehr stark an der verwendeten Hardware des Rechners orientiert sind, war eine Programmierung in einer Hochsprache nicht möglich. In einer anderen Umgebung muß es ohnehin neu erstellt werden. Sein geringer Umfang und seine einfache Funktion vereinfachen dies natürlich erheblich.

In einem Environment ohne Multitasking wäre die Realisierung einer solchen Echtzeitverwaltung etwas aufwendiger ausgefallen, aber dennoch möglich. Es hätte dann ein Zeitgeberbaustein eingesetzt werden müssen, der programmgesteuert zu den jeweiligen Abtastzeitpunkten eine Prozeßumschaltung als Interruptservice durchführt. Solange die Zahl der zu verwaltenden Prozesse gering ist, bringt diese Vorgehensweise keine grundsätzlichen Probleme mit sich, ist aber wesentlich uneleganter und schwieriger zu einem anderen Rechner zu portieren.

Ein anderer Punkt, welcher der Vorausplanung bedarf, ist die Art und Weise, wie Parameter vom Steuerprogramm zum Regelprogramm und zurück übertragen werden sollen. Im vorliegenden System wurde festgelegt, daß alle Parameter, die von der Regelung benötigt

werden, auch im Adressraum des Regelprogrammes liegen müssen. Dann kann das Steuerprogramm terminiert werden, ohne daß durch die Aufgabe des zugehörigen Speicherbereiches dem Regler seine Datenbasis entzogen wird. Es ist also möglich, das Steuerprogramm zu verlassen, während die Regelung arbeitet und gegebenenfalls nach Ausführen anderer Dienstprogramme wieder in das Steuerprogramm zurückzukehren. Diese Aufgabenteilung in einen Vordergrund und einen Hintergrund ergänzt die Leistungsfähigkeit eines solchen Systems erheblich.

Der Zugriff auf Daten innerhalb des Adressraumes des Echtzeitprogramms durch das Steuerprogramm wird am einfachsten und schnellsten durch "shared memory"-Verwendung ausgeführt. In der Programmiersprache Pascal können hierzu Pointer verwendet werden, die auf die jeweils erforderlichen Datenstrukturen zeigen.

7. 2. Blockorientierte Regelung

Eine vielfältig in der Technik, insbesondere auch in der Regelungstechnik verwendete Darstellungsform des Verhaltens dynamischer Systeme bedient sich Blöcke, die untereinander verbunden werden. Die Blöcke selbst definieren dabei ein Eingangs-Ausgangsverhalten, und die Verbindungen zwischen den Blöcken stellen den Signalfluß bildhaft dar. Solche Darstellungen sind übersichtlich, weil sie sich auf das Wesentliche konzentrieren und die Struktur des dynamischen Systems klar zeigen.

Einen durchaus vergleichbaren Ansatz kann man auch machen, um zu einer Form einer blockorientierten Regelung zu gelangen. Hierzu wird gedanklich eine Regelstruktur in ihren Blöcken und Verbindungen gebildet und dann als Grundlage zur Konfiguration eines

Rechnerprogramms eingesetzt.

Man hat dabei prinzipiell drei verschiedene Grundtypen von Blöcken zu unterscheiden. Es gibt Datenquellen, also Blöcke, die keinen Eingang innerhalb eines Programms haben, sondern von außen mit Werten versorgt werden müssen. Hierzu gehören die Analog-Digital-Wandler, die in einem Echtzeitprogramm zum Messen von Prozeßgrößen verwendet werden. Für die gelesenen Werte ist das Programm nur insoweit verantwortlich, als es den Lesevorgang initiieren muß. Die ausgangsseitigen Werte dieses Blockes entstehen aber ohne weiteres Zutun des Rechners. Eine zweite typische Datenquelle sind Sollwert oder Konstantenblöcke. Bei ihnen wird von außerhalb des Echtzeitprogramms, nämlich vom Steuerprogramm, ein Versorgungswert erzeugt. Das Echtzeitprogramm benutzt diesen Wert lediglich, kümmert sich aber nicht um seine Herkunft.

Ein anderer wichtiger Blocktyp stellt eine Daten-senke dar. Dies sind in erster Linie die Digital-Analog-Wandler, die der Erzeugung von Stellgrößen dienen. Allgemeiner kann man aber auch noch solche Blöcke zu dieser Gruppe zählen, die ausgangsseitig an keinen weiteren Block angeschlossen sind. Offene Strukturen dieser Art sollten auf keinen Fall verboten werden, da sie, wie sich gezeigt hat, ein wichtiges Hilfsmittel zu strukturellen Eingriffen in Regelkreise darstellen.

Die Mehrzahl der Blöcke aber wird zugleich über Ein- und Ausgänge verfügen und daher zur dritten Gruppe der datenbearbeitenden Blöcke gehören. Ihr wichtigster Vertreter ist der Regler, dessen Eingangs-Ausgangs-verhalten von einer Differenzengleichung beschrieben wird. Andere sind Summationspunkte, Begrenzer, Multiplizierer usw.

Ausgehend von diesem Grundgedanken einer block-orientierten Regelung wurden ein Echtzeitprogramm und

ein interaktives Steuerprogramm geschaffen, welche sehr allgemeine Regelkreisstrukturen zu realisieren erlauben. Struktur- und Blockdefinitionen werden im Steuerprogramm DDC vom Anwender durchgeführt [H4, W4]. Sobald diese komplett sind, können sie dem Echtzeitprogramm DDCRT übergeben werden und dort als Regelung ablaufen.

Bild 7.2.1 zeigt eine Liste aller im Augenblick vorhandenen Blocktypen des Programms. Zur Strukturdefinition bedient man sich der jeweiligen Blocktypkurzbezeichnungen wie SW für Sollwertblock, RE für Regler oder AD für Analog-Digital-Wandler.

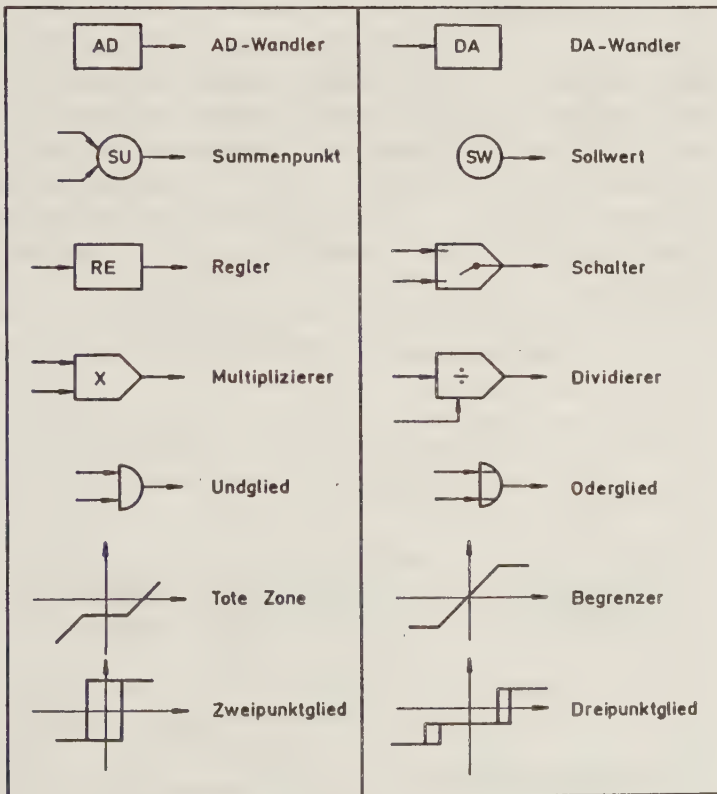


Bild 7.2.1 Liste der vorhandenen Blocktypen

Von sehr kleinen Anwendungen abgesehen, wird es immer nötig sein einen Blocktyp mehr als ein einziges Mal zu benutzen. Eine sehr direkte Unterscheidungsmöglichkeit ist in einer wahlfreien Numerierung zu sehen. Die Blöcke könnten dann etwa RE3, AD2 oder SU5 heißen. Die Verbindungen der Blöcke untereinander bestimmen dann die Regelkreisstruktur und damit den Signalfluß. Am einfachsten ist es, jeweils zu einem Block anzugeben, von welchen anderen Blöcken er seine Eingangssignale erhält. Es bedeuten dann

RE6, SU3

oder

SU3, AD2, SW1 ,

daß der Regler RE6 als Eingangssignal den Ausgang des Summenpunktes SU3 verwendet, und daß der Summenpunkt SU3 als Eingangssignal die Ausgangswerte aus den Blöcken AD2 und SW1 erhält. Bild 7.2.2 zeigt ein ganz einfaches Beispiel eines Regelkreises in blockorientierter Form. Die erforderliche Strukturdefinition lautet dann

SU1, SW1, AD1

RE1, SU1

DA1, RE1 .

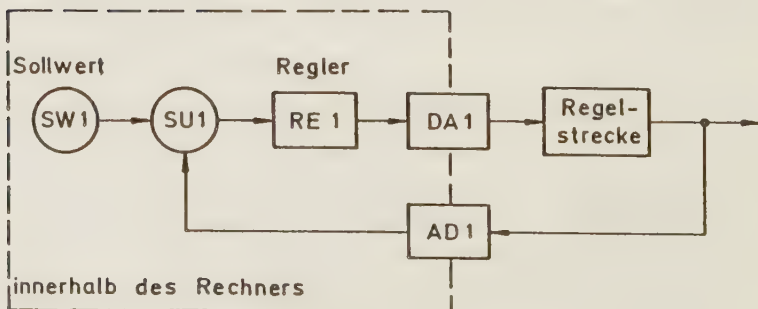


Bild 7.2.2 einfaches Beispiel eines Regelkreises in blockorientierter Form

In dieser Form, aber nicht notwendigerweise in der angegebenen Reihenfolge, würde man die Regelkreisstruktur für das Programm DDC eingeben. Die passende Reihenfolge der Eingabezeilen und damit auch die erforderliche Bearbeitungsreihenfolge wird durch das Programm selbständig durch einen sukzessiven Eliminationsalgorithmus bestimmt. Dieser geht davon aus, daß nur solche Blöcke bearbeitet werden können, deren Eingangssignale zum jeweiligen Abtastschritt schon vorliegen. Eventuelle arithmetische Schleifen werden dabei erkannt und abgelehnt. Es ist fast immer möglich, durch eine strukturelle Änderung arithmetische Schleifen zu vermeiden. In dem Beispiel zeigt sich auch sehr deutlich, daß die Regelstrecke selbst kein Teil der Konfigurierung des Programms ist, da sie ja außerhalb des Rechners liegt. Ferner ist das sonst übliche Minuszeichen am Summenpunkt von der Rückführung des Ausgangs der Regelstrecke weggelassen. Es ist ein Parameter des Blockes SUL und muß wie alle anderen Parameter der Blöcke in einer Spezifikationsphase angegeben werden. Diese Blockspezifikationen können in beliebiger Reihenfolge sofort oder auch erst zu einem späteren Zeitpunkt erfolgen. Spätestens vor dem Starten einer Regelung müssen aber alle Blöcke spezifiziert sein.

Die wesentlichsten Eigenschaften des Programms DDC und des zugehörigen Echtzeitprogramms DDCRT sind folgende :

- Es sind keine besonderen Mikrorechnerkenntnisse nötig, da das Programm interaktiv arbeitet.
- Je nach Blocktypenverteilung können knapp 100 Blöcke verwendet werden.
- Die Abtastzeit läßt sich in Vielfachen von 16,67 msec einstellen.

- Zur Abschätzung der Rechenzeit kann man sich an ca. 25 - 30 msec für einen PID-Regler einschließlich der erforderlichen AD-, DA-Wandler, Summenpunkt und Sollwert orientieren, wenn man voraussetzt, daß Realarithmetik in Softwareform verwendet wird. Mit einer hardwaremäßigen Unterstützung durch einen Am9511 Chip darf man mit einer Geschwindigkeitsteigerung um etwa den Faktor 4 bis 5 rechnen.
- Das Definieren und Ändern von Regelkreisstrukturen oder von Parametern in den vorliegenden Blöcken kann im "on-line"- und "off-line"-Betrieb erfolgen.
- Für einfache Anwendungen ist ein PID-Reglerentwurf eingebaut, sodaß dieser häufige Reglertyp sehr leicht zugänglich wird.
- Weitere Reglerentwürfe stehen in anderen Programmen des Systems zur Verfügung und können über Dateien in das Programm übertragen werden. Jeder einzelne Reglerblock kann einen Regler von maximal 8. Ordnung aufnehmen. Falls einmal eine noch höhere Ordnung erforderlich sein sollte, kann dies durch eine Reihenschaltung realisiert werden.
- Das Anwenderinterface ist in Menuetechnik aufgebaut und kommandoorientiert mit vielfältigen Listfunktionen und einem "On-Line"-Manual.
- Eine Speicherung der erzeugten Strukturen kann in Dateien erfolgen. Dies gilt auch, wenn sie noch unvollständig sind.
- Die Parameterübergabe kann jeweils sofort oder durch ein Kommando synchronisiert erfolgen. Im letztgenannten Fall werden mehrere Änderungen gleichzeitig gültig.

- Zur stoßartigen Umschaltung von verschiedenen Reglertypen können die in den Differenzengleichungen gespeicherten vorherigen Werte der Stellgröße und des Regelfehlers von einem Regler in einen anderen übertragen werden.
- Zu Auswertungszwecken können Signalwerte aufgezeichnet werden. Die Triggerung kann mit internen oder externen Signalen oder auch von Hand erfolgen. Um die anfallenden Datenmengen gegebenenfalls zu reduzieren, kann auch nur jeder i-te Abtastschritt mit einer Signalspeicherung verknüpft werden.
- Das Echtzeitprogramm arbeitet auch ohne Steuerprogramm, sodaß jederzeit mit anderen Programmen des Systems neue Regler entworfen oder graphische Darstellungen bearbeitet werden können.

Ein aufwendigeres Anwendungsbeispiel fand die blockorientierte Regelung bei einer Simulation des Verhaltens des Azimutteils eines 13.5 m Antennenreflektors [A3]. Die Regelstrecke wurde mit einem Analogrechner nachgebildet. Als Dimensionierungsgrundlage dienten Daten, die von der Firma Krupp Industrie- und Stahlbau zur Verfügung gestellt wurden. Die vorgeschlagene Regelstruktur wurde im Wesentlichen übernommen [B4]. Bild 7.2.3 zeigt den Aufbau des Reglers in der blockorientierten Form. Er besteht aus den beiden Reglern RE1 und RE2 zur Positionsregelung und in einer Kaskade unterlagerten Drehzahlregelung. Zusätzlich wird die Differenzdrehzahl über einen weiteren Regelkreis mit dem Regler RE3 eingesetzt, um die ständige Vorspannung der beiden Antriebsmotoren auch dynamisch aufrechtzuerhalten. Die statisch wirksame Vorspannung des Antennenantriebs wird über den Block SW3 erzeugt und über Summationsstellen auf die beiden Stellgrößen M1 und M2 aufgeschaltet und damit zu den Stellmotoren weitergeleitet.

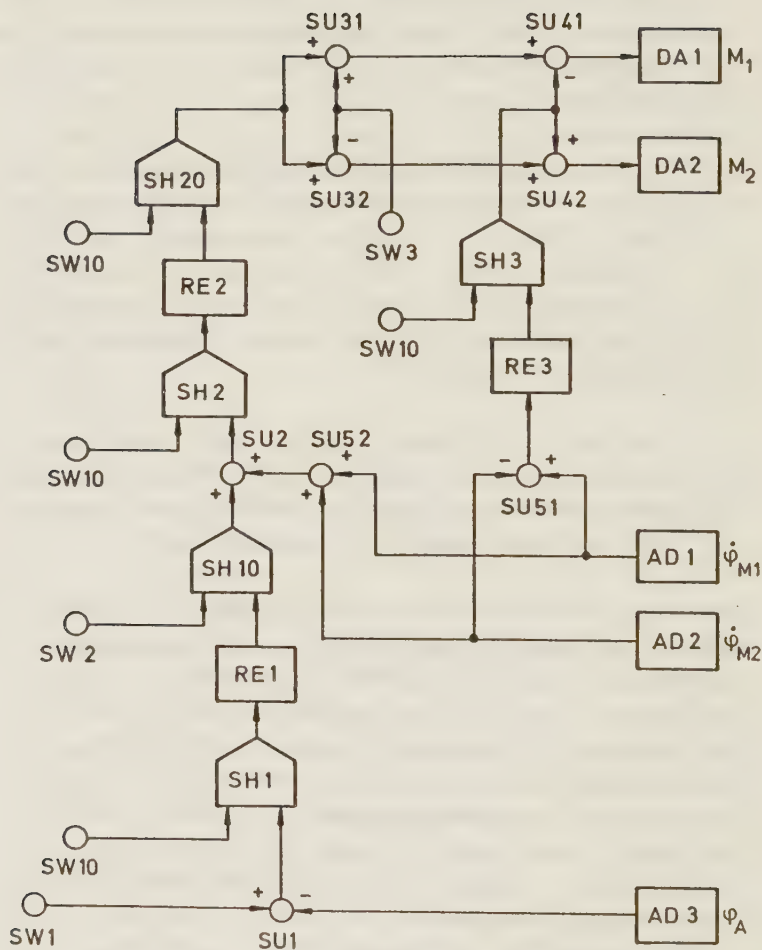


Bild 7.2.3 blockorientierte Regelung des Azimutteils eines 13.5 m Antennenreflektors

7. 3. Zustandsregler

In Kapitel 6. wurden Programme für Entwürfe von Zustandsreglern und Beobachtern vorgestellt. Um diese auch unmittelbar anwenden zu können, mußten ein Echtzeitprogramm ZEBRE und das dazu passende Steuerprogramm ZEBRA geschaffen werden [K2]. Über das Steuerprogramm kann eine Konfigurationsdatei gelesen und ausgewertet werden. Falls alle darin enthaltenen Teile einer Zustandsregelung vorliegen, d. h. wenn die Regelung vollständig entworfen wurde und keine erforderlichen Teile fehlen, werden die Matrizen in das Echtzeitprogramm übernommen. Anschließend bzw. nach Festlegen der zu verwendenden analogen Ein- und Ausgänge kann das Echtzeitprogramm gestartet werden, und die Regelung arbeitet. Interaktiv können während der Regelung Sollwerte verändert werden. Ein Anhalten der Regelung und ein Austauschen des Reglers ist jederzeit möglich. Für eine Untersuchung der erzielten Eigenschaften der Regelung müssen Signalwerte in entsprechende Dateien abgelegt werden. Es können hierbei sowohl gemessene Ausgangsgrößen der Regelstrecke und Stellgrößen des Reglers als auch interne Rechengrößen wie beobachtete Streckenzustände oder Störgrößen angesprochen werden. Weiterhin ist die Definition einer Triggerbedingung erforderlich. In den meisten Fällen wird man die Aufzeichnung der Signale mit einer Sollwertänderung beginnen oder aber von Hand triggern.

Bevor jedoch eine entworfene Regelung in diesem Echtzeitprogramm verwendet werden kann, muß eine Aufbereitung der Matrizen in einem Vorbereitungsprogramm ZEVOR durchgeführt werden. Die Regler- und Beobachterentwürfe benutzen alle Regler- oder Beobachterentwurfssysteme, die von dem ursprünglichen System abweichen können. Bei dem Reglerentwurfssystem wird dies typischerweise dann der Fall sein, wenn eine PI-Regelung gewünscht wurde, und das Beobachterentwurfssystem wird vom ursprünglichen System verschieden sein,

wenn eine Störgrößenaufschaltung im Beobachter oder ein Störbeobachter konfiguriert wurde. Außerdem ist der Zustandsvektor eines Beobachters erst nach einer Transformation für die Regelung geeignet, sofern nicht ein Identitätsbeobachter eingesetzt wurde.

Zunächst wird die Transformationsmatrix $\underline{P}$ des Beobachters gebildet durch

$$\underline{P} = \underline{T}^{-1} \quad \text{bei vollständigen Beobachtern}$$

$$\underline{P} = \begin{bmatrix} \underline{T} \\ \underline{C} \end{bmatrix}^{-1} \quad \text{bei reduzierten Beobachtern}$$

Die Matrix $\underline{C}$ ist hier die Ausgangsmatrix der ursprünglichen Regelstrecke. Damit die erforderliche Transformation nicht ständig "On-line"-Betrieb gerechnet werden muß, ist sie entweder mit den anderen Matrizen des Beobachters oder mit den Reglermatrizen zusammenzufassen [K4]. Der erstgenannte Weg, das Berücksichtigen der Transformation in den Matrizen des Beobachters ist allerdings nur bei vollständigen Beobachtern anwendbar. Die Konventionen zum Zusammenbau der in der Regelung auftretenden Vektoren sind ohne Beschränkung der Allgemeingültigkeit :

- Der Beobachterzustandsvektor besteht aus einem ersten Teilvektor für die geschätzten und gemessenen Zustandsgrößen der Regelstrecke und gegebenenfalls aus einem zweiten Teilvektor für die geschätzten Störgrößen.
- Der zur Störgrößenaufschaltung verwendete Störvektor besteht aus einem Teilvektor für die gemessenen Störgrößen und einem darangehangenen Teilvektor für die geschätzten Störgrößen.

Für vollständige Beobachter ergeben sich die umgerechneten Matrizen des Beobachters - Index P - zu

$$\underline{D}_P = \underline{P} \underline{D} \underline{P}^{-1}$$

$$\underline{E}_P = \underline{P} \underline{E}$$

$$\underline{G}_P = \underline{P} \underline{G} = \underline{B}$$

$$\underline{B}_z P = \underline{P} \underline{B}_z .$$

Hiermit wird nachträglich eine Transformation des Beobachters zu einem Identitätsbeobachter durchgeführt. Notwendig ist diese Transformation nicht. Statt ihrer kann die Matrix $\underline{P}$ auch mit der Reglermatrix

$$\underline{F}_x P = \underline{F}_x \underline{P}$$

zusammengefaßt werden. Für reduzierte Beobachter ist sogar nur dieser Weg vorhanden.

Enthält der Beobachter einen Anteil zur Schätzung von Störgrößen, so müssen die Reglermatrix $\underline{F}_x$ und die Störgrößenaufschaltmatrix $\underline{F}_z$ mit der Transformationsmatrix $\underline{P}$ gemeinsam behandelt werden [K4]. Mit einem Vektor aus den Beobachterzuständen $\underline{v}$ und den Streckenausgängen $\underline{y}$ muß für den Vektor aus Streckenzustand $\underline{x}$ und beobachteter Störgröße $\underline{z}$

$$\begin{bmatrix} \underline{x} \\ \underline{z} \end{bmatrix} = \underline{P} \begin{bmatrix} \underline{v} \\ \underline{y} \end{bmatrix}$$

gelten. Bei der Bildung der Stellgröße $\underline{u}$ tritt neben weiteren Größen der Term

$$- \underline{F}_x \underline{x} + \underline{F}_z \underline{z} = \begin{bmatrix} -\underline{F}_x & \underline{F}_z \end{bmatrix} \begin{bmatrix} \underline{x} \\ \underline{z} \end{bmatrix}$$

auf. Setzt man den zuvor genannten Zusammenhang ein und teilt die Matrix $\underline{P}$ in vier Teilmatrizen auf, die zu den Dimensionen der Vektoren passen, so erhält man

$$- \underline{F}_x \underline{x} + \underline{F}_z \underline{z} = \begin{bmatrix} -\underline{F}_x & \underline{F}_z \end{bmatrix} \begin{bmatrix} \underline{P}_a & \underline{P}_b \\ \underline{P}_c & \underline{P}_d \end{bmatrix} \begin{bmatrix} \underline{v} \\ \underline{y} \end{bmatrix}.$$

Durch Ausmultiplizieren erhält man die transformierten Matrizen der Regelung zu

$$- \underline{F}_x \underline{x} + \underline{F}_z \underline{z} = \quad (- \underline{F}_x \underline{P}_a + \underline{F}_z \underline{P}_c) \underline{v} \\ + (- \underline{F}_x \underline{P}_b + \underline{F}_z \underline{P}_d) \underline{y} .$$

Man erkennt deutlich, daß die Störgrößenaufschaltung und die Zustandsrückführung den Beobachterzustand und die gemessenen Ausgangsgrößen verwenden. Zuletzt muß noch eine Aufteilung der Rückführmatrix $\underline{F}_x$ erfolgen, falls die Regelung eine PI-Struktur besitzt.

7. 4. Regelung mit Differenzengleichungen

Zur direkten digitalen Regelung mit Reglern, die einer rekursiv zu jedem Abtastzeitpunkt auszuwertenden Differenzengleichung genügen, wurde ein passendes Echtzeitprogramm ADR erstellt, welches den "On-line"-Betrieb solcher Regler ermöglicht. Vorausgesetzt wurde hierbei, daß der angeschlossene Prozeß über die vorhandenen Digital-Analog- und Analog-Digitalwandler zugänglich ist.

Die Struktur des Regelkreises wurde bereits in Kapitel 6.1. behandelt. Sie besteht im Falle des Echtzeitprogramms aus einem Vorfilter $P(z)$, einem Regler $R(z)$ und einem zusätzlichen Filter $Q(z)$ im Rückkopplungszweig, falls der bereits erwähnte seltene Fall vorliegt, daß instabile Polstellen des Vorfilters durch Nullstellen des Regelkreises kompensiert werden sollen. Bild 7.4.1 zeigt einen solchen digitalen Regelkreis.

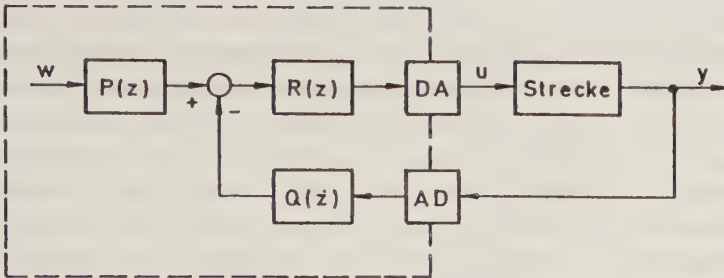


Bild 7.4.1 Regelkreis zur direkten digitalen Regelung

Als Regler kommen hier alle Entwürfe in Betracht, die von den Entwurfsprogrammen aus Kapitel 6.1. berechnet wurden oder auf irgendeine andere Weise in Form einer Datei vorliegen. Die zuletzt genannte Methode, völlig frei wählbare Übertragungsfunktionen als Regler zu verwenden, erlaubt es, auch solche Regler zu einzusetzen, die nicht von dem hier beschriebenen Softwaresystem ermittelt wurden. Die maximale Ordnung ist zur Zeit mit 10 festgelegt und gibt damit einen erheblichen Raum auch für sehr aufwendige Regler. Da die zu verwaltenden Parameter nicht besonders umfangreich sind, stellt die Reglerordnung hier keine Einschränkung dar. Gemeinsam mit der Regelung kann eine Aufzeichnung der während des Betriebs anfallenden Signalwerte erfolgen. Diese stehen nach der Meßwert-erfassung weiteren Untersuchungen zur Verfügung. Die Meßwertaufzeichnung selbst kann entweder manuell oder durch eine positive oder negative Flanke eines Signales getriggert werden.

Für einfache Anwendungen dürfen die Übertragungs-funktionen $P(z)$ und/oder $Q(z)$ auch entfallen. Sie werden dann automatisch durch eine 1 ersetzt, was einem unmittelbaren Durchschalten des Sollwertes w bzw. des gemessenen Istwertes y auf den Summenpunkt entspricht.

Nach außen hin steht die Regelung in einem komfor-tablen Steuerprogramm an den Anwender zur Verfügung. Alle Teile des Echtzeitprogramms sind interaktiv

zugänglich und erleichtern die Handhabung durch eine Reihe von Funktionen. Plausibilitätstest, Zustandsanzeigen und nötigenfalls Fehlermeldungen durch Klartext verhindern Fehler frühzeitig. Wie in allen anderen Steuerprogrammen, kann auch hier das Echtzeitprogramm alleine weiterarbeiten, wenn das Steuerprogramm verlassen wird, um mit einem der anderen Programme des Systems einen neuen Regler zu entwerfen, gemessene Signale zu untersuchen oder andere Arbeiten durchzuführen. Bei einem erneuten Starten des Steuerprogramms wird die arbeitende Regelung erkannt, und die im Echtzeitprogramm vorliegenden Parameter sind wieder zugänglich.

Eine wichtige weitere Eigenschaft des Echtzeitprogramms ist in einer Vorkehrung gegenüber einem Effekt zu sehen, der in der Literatur oft als "Integrator Windup" bezeichnet wird [U2]. Er entsteht dadurch, daß für längere Zeit der integrale Anteil eines Reglers durch die Regelabweichung am Summenpunkt unzulässig angeregt wird. Denkt man sich den Regelalgorithmus in analoger Technik ausgeführt, so entspräche diesem Vorgang ein ständiges Aufintegrieren der Regelabweichung. In einer analogen Schaltung ist ein solcher Vorgang nicht möglich, da spätestens bei Erreichen der Versorgungsspannung das Ausgangssignal eines Integrators durch Nichtlinearität begrenzt wird. Mit einem Rechner realisiert, wird der vergleichbare Vorgang der Summenbildung weitergehen, da ihm durch den Bereich der darstellbaren Zahlenwerte erst bei erheblich größeren Werten eine Begrenzung entgegensteht. Eine verhältnismäßig einfache, aber sehr wirkungsvolle Methode besteht in der Rückführung der begrenzten Stellgrößen $u(k)$, die an die Regelstrecke weitergegeben wurden, in die Folge der gespeicherten Stellgrößen $u(k-i)$. Jede einzelne begrenzte Stellgröße wird dann mit durch den Signal-speicher der Differenzengleichung durchgeschoben. Es gibt allerdings Fälle, in denen hiermit zufriedenstellende Resultate nicht erzielt werden können. Der "windup"-Effekt ist jedoch sicher ausgeschlossen. Für

ein einsatzfertiges Standardprogramm ist diese einfache Lösung aber wirkungsvoll. Das Verfahren wurde an einer Vielzahl von Regelstrecken verwendet und hat gute Ergebnisse geliefert.

8. Simulation von Regelkreisen

Normalerweise wird man eine entworfene Regelung nicht ohne weitere Überprüfung in Betrieb nehmen, sondern zuvor untersuchen, ob die gewünschten Eigenschaften auch wirklich erzielt wurden. Dies ist besonders wichtig, wenn beim Entwurf des Reglers eine linearisierte Regelstrecke angenommen wurde. Der Regler muß dann in seinem echten Anwendungsfall eine ganz andere Regelstrecke regeln, als die, für die er entworfen wurde. Ob eine solche Vorgehensweise überhaupt zulässig ist, sei dahingestellt. Sie wird aber oft beschrieben und stellt bei geringfügigen Nichtlinearitäten sicherlich einen brauchbaren Weg dar.

Eine recht tiefgreifende Analyse der Regelung ist durch eine Simulation möglich [K5]. Im Einzelfall können hierbei auch Eigenschaften der Regelstrecke simuliert werden, die für den Entwurf nicht herangezogen wurden. Man läßt also den Regler zusammen mit einem durch ein Programm realisierten Modell der Regelstrecke arbeiten und kann hierzu sogar eine genauere Modellierung der Strecke wählen, als sie zunächst verwendet wurde. Eine qualifiziertere Abschätzung der erzielten Eigenschaften der Regelung ist somit möglich.

Nachfolgend werden verschiedene Möglichkeiten zur Simulation dynamischer Systeme mit vorgefertigter Software beschrieben, die für die eben genannten Untersuchungen herangezogen werden sollte.

8. 1. Simulation im Zustandsraum

Ähnlich wie durch das Programm ZEBRA und sein Echtzeitprogramm ZEBRE eine digitale Zustandsregelung realisiert wird, so ist durch SIMZ eine Simulation einer solchen Regelung möglich [K2]. Das Simulationsprogramm besitzt allerdings eine Reihe von Ergänzungen und Funktionsänderungen gegenüber der Regelung in Echtzeit.

Der grundlegende Unterschied besteht zunächst einmal darin, daß es kein Echtzeitprogramm gibt, sondern die gesamten Berechnungen in einem einzigen Programm durchgeführt werden. Die Zeit ist in der Simulation nur eine Rechengröße. Irgendwelche Wartebedingungen bis zum Zeitpunkt eines nächstfolgenden Abtastschrittes sind nicht erforderlich. Vielmehr werden alle Berechnungen unmittelbar nacheinander ausgeführt, sodaß die zur Simulation erforderliche Zeit, die nicht mit der gerechneten Simulationszeit verwechselt werden darf, nur die Rechenzeit für den Algorithmus selbst und natürlich die Zeit zur Ausgabe der Simulationsergebnisse umfaßt. Vielfach wird diese Ausgabe in der Form eines Signalverlaufs auf dem Sichtgerät angezeigt werden müssen. Die Rechenzeit für eine solche Darstellung kann im Einzelfall die zur Simulation benötigte Zeit überschreiten. In jedem Fall wird die gesamte zur Simulation nötige Zeit größer sein als zur Regelung, da nebenbei auch noch die Regelstrecke mitgerechnet werden muß, und da jede sinnvolle Datenausgabe eines Simulationsprogramms mehr Zeit kostet als eine Ausgabe des Stellvektors an die DA-Wandler. Weiterhin wird die Transformationsmatrix P des Beobachters etwas aufwendiger behandelt. Neben der Einbeziehung der Transformation in die übrigen Matrizen, wie sie bereits in Kapitel 7.3. dargestellt wurde, kann sie auch unmittelbar auf den Beobachterzustand angewendet werden, sodaß die vom Beobachter geschätzten Zustandsgrößen zu Vergleichszwecken mit den

gerechneten Zustandsgrößen der Regelstrecke in rücktransformierter Form verfügbar sind. Falls der Beobachter von reduzierter Ordnung ist, wird natürlich der simulierte Ausgangsvektor der Strecke gemeinsam mit dem Zustandsvektor des Beobachters rücktransformiert.

Neben der Simulation eines kompletten Regelkreises ist es auch möglich, das Verhalten der Regelstrecke allein, nur der Regelstrecke mit Beobachter oder auch der Regelstrecke mit Regler aber ohne Beobachter zu simulieren. Damit kann unmittelbar Information auch über die verschiedenen Zwischenstufen eines Reglerentwurfs gewonnen werden.

In Bild 8.1.1 ist das Resultat einer Simulation einer Regelstrecke dargestellt. Es handelt sich um ein diskretisiertes System 2. Ordnung bestehend aus zwei hintereinandergeschalteten PT_1 -Gliedern mit Zeitkonstanten von 1 bzw. 0,5 Sekunden. Die Anfangswerte des Beobachters wurden zu Null gewählt, während die Regelstrecke von Null verschiedene Anfangswerte besaß. Dargestellt sind jeweils ein Zustand der Regelstrecke zusammen mit ihrer rücktransformierten Schätzung durch den Beobachter. Bei einer Abtastzeit von 0,1 Sekunden lagen die Eigenwerte des Beobachters bei

$$z_1 = 0,74$$

$$z_2 = 0,67 \quad .$$

In der graphischen Darstellung wurden die Zustandsgrößen der Regelstrecke kontinuierlich, die des Beobachters stufenförmig gezeichnet, um die realen Verhältnisse des diskreten Beobachters an der kontinuierlichen Strecke besser darzustellen.

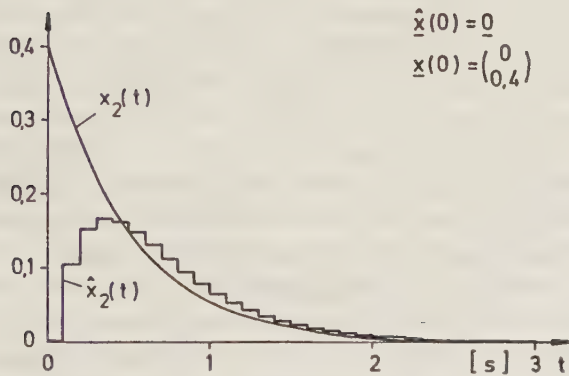
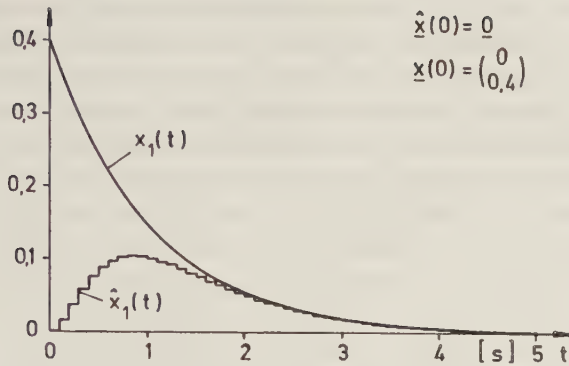


Bild 8.1.1 Simulation einer Regelstrecke und eines Beobachters mit unterschiedlichen Anfangsbedingungen

8. 2. Simulation mit Differenzengleichungen

In Kapitel 8.1. wurde zur Simulation eine Zustandsraumdarstellung des Regelkreises vorausgesetzt. Eine andere wichtige Beschreibungsform dynamischer Systeme sind Übertragungsfunktionen. Unabhängig von der

Möglichkeit, die eine Form in die andere umzurechnen, ist es notwendig, auch unmittelbar anhand gegebener Übertragungsfunktionen eine Simulation durchzuführen. Diese Aufgabe wird durch ein Simulationssystem SIMG gelöst, welches direkt die Eingabe von kontinuierlichen oder diskreten Übertragungsfunktionen erlaubt.

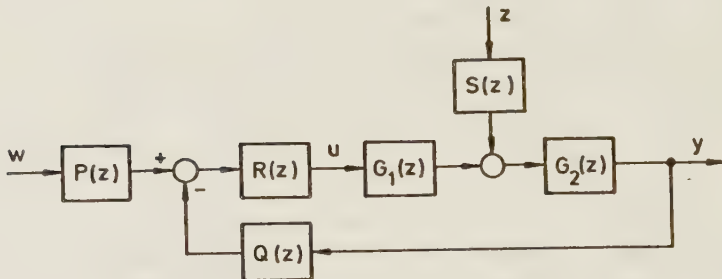


Bild 8.2.1 Regelkreisstruktur der Simulation

Um eine möglichst weitgehende Analyse des Verhaltens eines Regelkreises oder auch nur einer Regelstrecke zu erlauben, wurde die in Kapitel 6.1. für Reglerentwürfe vorausgesetzte Regelkreisstruktur nochmals erweitert [H7]. Bild 8.2.1 zeigt diese zugrundeliegende Struktur für den diskreten Fall. In einem konkreten Anwendungsfall ist es natürlich zulässig, von einer vereinfachten Struktur auszugehen, wenn nicht alle dargestellten Blöcke erforderlich sind. Wie bereits in den Kapiteln 6.1. und 7.4. angeführt, ist das Rückkopplungsfilter $Q(z)$ nur unter besonderen Bedingungen erforderlich. Eine gleichzeitige Aufschaltung von veränderlichen Führungsgrößen w und Störungen z erscheint ebenfalls nur sehr selten erforderlich, da sich in einem solchen Fall das Verhalten der Regelung auf beide Situationen überlagert. Zumindest im Prinzip könnte also jeweils eine der Übertragungsfunktionen $P(z)$ oder $S(z)$ entfallen. Die Regelstrecke $G(z)$ wurde in zwei Teilübertragungsfunktionen $G_1(z)$ und $G_2(z)$ aufgeteilt. Dadurch wird es möglich, über den Eingriffspunkt der Störgröße frei zu

verfügen, um die Simulation den physikalischen Gegebenheiten der Regelstrecke besser anzupassen. Als ein letzter Sonderfall sei noch erwähnt, daß es auch möglich ist, das Verhalten der Regelstrecke ohne einen Regler zu simulieren. Insgesamt ergibt sich aus all diesen Möglichkeiten eine sehr große Vielzahl von Regelkreiskonfigurationen und Betriebssituationen, die in interaktiver Form den Wünschen des Anwenders entsprechend simuliert werden können.

Als Eingangsgrößen für die Simulation, d.h. als Sollwertsignal w oder als Störsignal z können neben den häufig verwendeten sprungförmigen Änderungen auch ein Impuls, ein PRBS-Signal, ein approximiertes weißes Rauschen oder ein Signal von einer entsprechenden Datei verwendet werden. Damit können nicht nur die Sprung- und Impulsantworten des Regelkreises simuliert werden, sondern auch das Betriebsverhalten mit Signalen, die repräsentativen Sollwert- oder Störungsverläufen entsprechen. Verwendet man PRBS-Signale oder Rauschen, und werden die simulierten Signale aufgezeichnet, so können sie für Korrelationsanalyse herangezogen werden. Unterstellt man, daß Mittelwert und Varianz eines angreifenden Störsignals bekannt sind, dann ist eine Bewertung der Regelung anhand der entsprechenden Eigenschaften der Regelgröße leicht durchführbar.

8. 3. Blockorientierte Simulation

Ein wichtiges Hilfsmittel bei der digitalen Simulation des Verhaltens dynamischer Systeme stellen blockorientierte Simulationsprogramme dar. Sie stellen eine Umsetzung von Blockschaltbildern in Programme dar und werden in allen Bereichen der Technik verwendet. Im Bereich der Regelungstechnik wird die Verwendung von Blockschaltbildern zur Kennzeichnung von Struktur und

Aufbau von Regelkreisen schon lange mit gutem Erfolg eingesetzt und ist daher weit verbreitet. Geringer Aufwand bei der Zeichnung von Hand und gute Übersichtlichkeit sind für blockorientierte Darstellungen kennzeichnend.

Dies legte schon frühzeitig in der Rechentechnik den Einsatz von Verfahren nahe, die sich ganz wesentlich auf der Blockorientierung abstützen. Eine sehr bekannte Simulationssprache für dynamische Systeme und zugleich eine der ältesten ist die Simulationssprache CSMP - Continuous System Modelling Program [F2, G2, S2, U1]. Die Leistungsfähigkeit blockorientierter Simulationssprachen wurde nicht zuletzt durch Programme wie CSMP demonstriert. Ein Nachteil von CSMP und einiger anderer Simulationssprachen ist ihre Beschränkung auf kontinuierliche Systeme. Zwar mögen diese in der Technik weitaus häufiger anzutreffen sein, doch ist vor allem für die Regelungstechnik der Verzicht auf zeitdiskrete Vorgänge nicht sinnvoll, da der Haupterfolg regelungstechnischer Forschung und Anwendung gerade im Einsatz von Rechnern, und damit in der Verwendung von diskreten Verfahren, zu finden ist. Mithin sollte ein Simulationsprogramm auch die Behandlung diskreter Systeme erlauben. Für normale regelungstechnische Aufgaben ist eine Ereignisorientierung wie in GASP [C1] von geringerer Bedeutung, sodaß auf sie oftmals verzichtet werden kann.

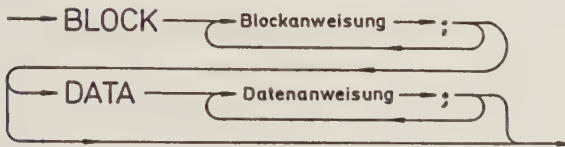
Programm:

SIMB → Bezeichner → ; → Programmblock → END.

Bezeichner:



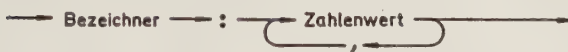
Programmblock:



Blockanweisung:



Datenanweisung:



Kommentare:



Bild 8.3.1 Syntaxdiagramm der Simulationssprache SIMB
(gekürzt)

Für das besprochene System wurde daher eine block-orientierte Simulationssprache SIMB entwickelt, deren Simulationsprogramme sowohl kontinuierliche als auch diskrete Regelkreiselemente umfassen darf [H3]. Ergänzt wurden weiterhin einige hybride Elemente, sodaß in Grenzen auch hybride Simulationen möglich sind. Das Konzept der Blockorientierung wurde überall streng eingehalten. Die etwas schwerfällige Programmierung mit CSMP aber konnte vermieden werden. Ein Simulations-

programm mit dem heutigen Stand der Technik sollte auch sprachlich strukturiert sein, wie man es etwa von Pascal her gewöhnt ist. Bild 8.3.1 zeigt das Syntaxdiagramm der Simulationssprache SIMB. Vor allem die völlige Aufgabe einer Zeilenorientierung und die Möglichkeit an jeder beliebigen Stelle innerhalb eines Programms Kommentare einzufügen, schafft in SIMB die Voraussetzung für sehr leicht lesbare Programme.

Ganz wesentlich wird die Programmdokumentation auch dadurch unterstützt, daß die Blöcke eines Anwenderprogramms nicht einem uneinsichtigen Nummernschema folgen müssen, sondern jeweils mit sinnfällig gewählten Bezeichnungen versehen werden können. Ein Regelstreckenausgang kann dann etwa "Ausgang" genannt werden, der Eingang der Strecke mag "Ventil" heißen, um auf die physikalische Natur der Stellgröße hinzuweisen, und die Regelabweichung, die Eingangsgröße des Reglers ist, könnte "Fehler" genannt werden. Bedient man sich dieser Sprachmittel, so gelangt man zu Simulationsprogrammen, die auch zu späteren Zeitpunkten noch leicht verständlich sind und damit ohne größeren Aufwand weiterverwendet werden können. Besonderer Wert sollte auch auf die Konstruktion eines leicht pflegbaren und ergänzbaren Übersetzers gelegt werden [C1].

Die zur Zeit in SIMB zugänglichen 27 Typen von Blöcken sind in Bild 8.3.2 aufgeführt. Allen Blöcken gemeinsame Eigenschaften sind :

- 1 bis 3 Eingänge, jeweils vom Typ des Blocks abhängig.
- 1 Ausgang, der gemeinsam mit dem Block den Namen trägt zur Kennzeichnung des entstehenden Signals.
- Die Typenbezeichnung ist keine einbuchstabige Abkürzung, sondern ein Wort, welches die Block-eigenschaft umreißt, z.B. "GAIN", "NOISE" oder "RELAY".

- Da die Blockerkennung anhand des Namens erfolgt, können mehrere gleiche Blocktypen aber mit verschiedenen Namen verwendet werden. Namensgleichheit auch bei verschiedenen Blocktypen ist natürlich ausgeschlossen.

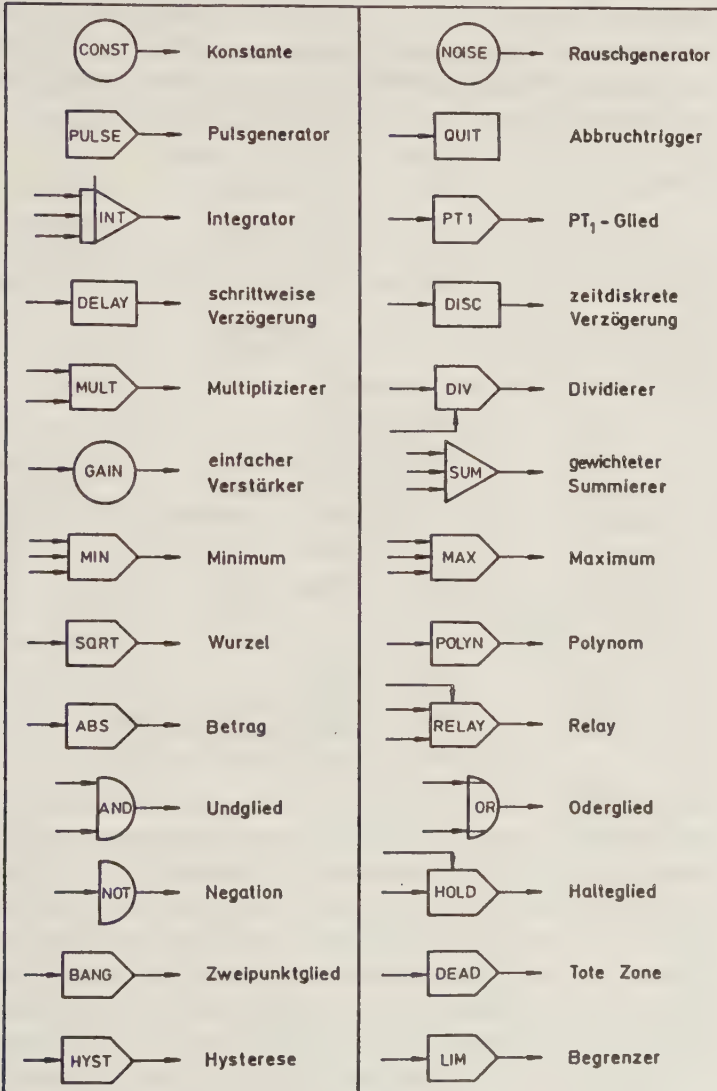


Bild 8.3.2 Blocktypen von SIMB

Nachfolgend ist ein Programmbeispiel für den einfachen Regelkreis in Bild 8.3.3 angegeben [H3]. Es handelt sich um einen Regelkreis aus einer Regelstrecke 2. Ordnung und einem PI-Regler. Über die Parameter des Blockes STÖRUNG kann wahlweise eine zusätzliche Störgröße zu einem beliebigen Zeitpunkt am Streckeneingang zur Wirkung kommen. Bild 8.3.4 zeigt das Signal am Streckenausgang mit und ohne Störung.

SIMB Beispiel ;

(*

Regelung einer PT2 Regelstrecke mit einem PI-Regler

*)

```
BLOCK { Blockdefinitionen }
      { Sollwert }
      SPRUNG      = CONST          ;
      { es folgt der Regler }
      FEHLER      = SUM  SPRUNG  AUSGANG  ;
      PTEIL       = GAIN  FEHLER          ;
      ITEIL       = INT  FEHLER          ;
      STELLGRÖSSE = SUM  ITEIL  PTEIL    ;
      { Störeingriff }
      STÖRUNG     = BANG  TIME          ;
      EINGANG     = SUM  STÖRUNG  STELLGRÖSSE  ;
      { Definition der Regelstrecke }
      MITTE      = PT1  EINGANG          ;
      AUSGANG     = PT1  MITTE           ;
```

```

DATA { Datenanweisungen }

SPRUNG      : 0.3                ; { Höhe }
FEHLER      : -1.0 , 1.0        ;
PTRIL      : 0.1                ; { P-Anteil }
ITEIL      : 0.0 , 1.0         ; { I-Anteil }
            { Anfangsbedingung, Verstärkungsfaktor }
STELLGRÖSSE : 1.0 , 1.0        ;
            { mit, ohne Störung, Zeitpunkt }
STÖRUNG     : 0.5 , 0.0 , 10.0 ;
EINGANG     : 1.0 , 1.0        ;
            { Anfangsbed., Verstärkung, Zeitkonstante }
MITTE      : 0.0 , 1.0 , 1.0 ;
AUSGANG     : 0.0 , 1.0 , 1.0 ;

```

END.

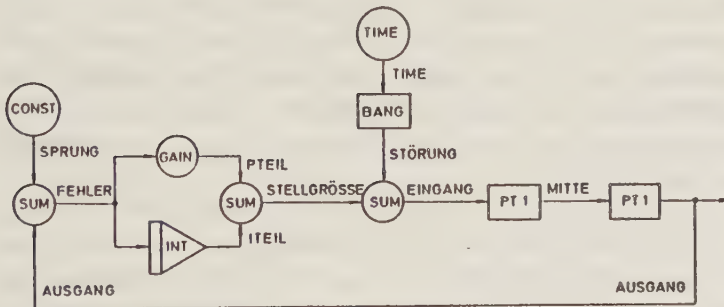


Bild 8.3.3 Beispiel eines Regelkreises mit
SIMB - Bezeichnern

Man erkennt unschwer, daß in den Blockanweisungen die Elemente der Regelkreisstruktur durch Angabe des Blocktyps und seiner Eingangsgrößen definiert werden. Die Parametrierung erfolgt in den Datenanweisungen für jeden definierten Block. Besonders hier empfiehlt sich eine starke Nutzung der Möglichkeit einer Kommentierung. Die Aufteilung eines Programms in einen strukturellen Teil und einen Datenteil ist auch in anderen Sprachen wiederzufinden und entspricht moderner Softwarearchitektur [C1].

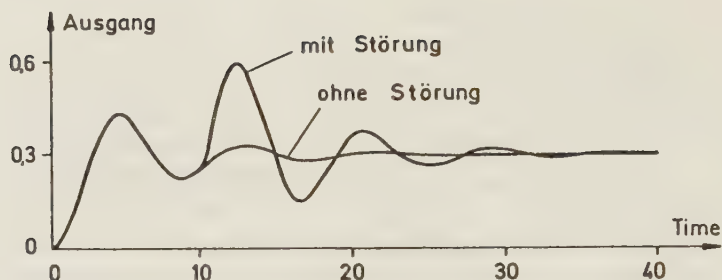


Bild 8.3.4 Ausgangssignal des simulierten Regelkreises im Beispielsprogramm

Der größte Unterschied zu anderen Simulationsprogrammen ist in der interaktiven Benutzerführung zu sehen. Die SIMB-Programme werden zwar normal editiert, dann aber durch das Simulationsprogramm selbst in Datentabellen übersetzt, sortiert und stehen danach sofort zur Verfügung. Der bei großen Simulationssystemen oft eingeschlagene Weg über Preprozessoren und anschließendes Compilieren wurde zugunsten einer einfacheren Anwendung für regelungstechnische Aufgaben verworfen [C1, U1].

In einem normalen CP/M - System mit 64 kBytes RAM reicht der Speicherplatz aus, um ca. 300 Blöcke zu behandeln. Damit sind bereits recht komplexe Strukturen simulierbar.

Eine recht aufwendige Anwendung fand das Programm bei der Simulation des Verhaltens des Azimutteils eines 13.5 m Antennenreflektors [A3]. Das mathematische, parametrisierte Modell der Regelstrecke wurde von der Firma Krupp Industrie- und Stahlbau zur Verfügung gestellt. Die im Original vorgesehene Regelstruktur mit 2 PI-Reglern zur Positions- und Drehzahlregelung und eines P-Reglers zur Differenzdrehzahlregelung wurde beibehalten [B4]. Zunächst wurden aus den Simulationsergebnissen einige wesentliche Kenngrößen entnommen, die zur Modellreduktion verwendet wurden. Für die anschließende Regelung eines analog realisier-

ten Prozeßmodells - vgl. Kapitel 7.2. - wurde dieses vereinfachte Modell eingesetzt.

8. 4. Simulation in Echtzeit

Zeitnormierung ist überall in der Simulationstechnik üblich bei analoger Simulation. Mit ihrer Hilfe können sehr langsame oder sehr schnelle Vorgänge in einem anderen, angenehmeren Zeitmaßstab ablaufen. Kriterium für die Auswahl der Simulationszeit ist häufig die Zeichengeschwindigkeit von Plottern oder Oszilloskopen. Bei einer digitalen Simulation ist die Zeit eine von vielen Rechengrößen. Die Dauer der Simulation steht in keinem unmittelbaren Zusammenhang zur mitberechneten Problemzeit, sondern wird einzig durch die zur Simulation notwendige Rechenzeit bestimmt.

Gelegentlich wird aber eine Simulation in Echtzeit gefordert; z.B. wenn ein Regler - analog oder digital - ein Prozeßmodell regeln soll, welches physikalisch nicht vorhanden ist oder aus Sicherheitsgründen nicht verwendet werden darf. Ein vielbeschrittener Weg ist es, den Prozeß mittels eines Analogrechners zu modellieren und dieses Modell zu regeln. Alternativ kann auch eine digitale Simulation des Prozesses mit einem Rechner in Echtzeit ausgeführt werden und dieses Modell dann mit dem gleichen oder einem zweiten Rechner geregelt werden. Der Vorteil einer solchen Vorgehensweise liegt in den häufig viel einfacheren und angenehmeren Benutzungseigenschaften der digitalen Simulation, welche die Notwendigkeit einer digitalen Echtzeitsimulation begründen [C1]. So ist es bei digitalen Simulationen zum Beispiel meist nicht erforderlich, den Prozeß zu normieren. Vielmehr können die Prozeßgrößen unmittelbar in ihren physikalischen Werten

benutzt werden. Bei der Analyse der erhaltenen Simulationsergebnisse spart man sich bei einer digitalen Simulation außerdem das Rückrechnen in die physikalischen Größen. Voraussetzung hierfür ist aber die Möglichkeit auch den Prozeß auf dem Rechner in Echtzeit zu simulieren. Der Sonderfall, daß Strecke und Regler zugleich in einem einzigen Programm realisiert und daher von der Echtzeit unabhängig sind, soll hier einmal außer acht gelassen werden, da er ja der normale Fall für eine Simulation ohne Echtzeitananschluß ist.

Für experimentelle Untersuchungen an kontinuierlichen oder diskreten Regelungen bietet die Echtzeitsimulation der Regelstrecke eine leicht und flexibel handhabbare Eingriffsmöglichkeit. Die Aufzeichnung der anfallenden Signale kann weitere Analysen unterstützen.

Ein ganz besonderer Vorteil einer solchen Simulation ist darin zu sehen, daß die Regelstrecke schrittweise von einer Näherungsversion, für die der Regler entworfen wurde, zu einer verfeinerten Version ergänzt werden kann, die auch Nichtlinearitäten der Strecke berücksichtigt. Treten mehrere solcher Besonderheiten auf, so kann man durch den Versuch ermitteln, welche von ihnen einen sehr großen und welche einen geringeren Einfluß auf das Verhalten der Regelung besitzen. Die so erhaltene Kenntnis über die Regelung kann dann in den Entwurf verbesserter Regler einfließen.

Ein grundsätzliches Problem aller Echtzeitsimulationen liegt in der erforderlichen Rechenzeit, welche die obere Grenzfrequenz beschränkt [G2]. Mit dem vorliegenden System, implementiert auf einem 8 Bit MP/M Rechner und ohne zusätzliche Hardware für Realarithmetik, gelten die bereits in Kapitel 7.2. angeführten Abschätzungen. Für ein System 2. Ordnung, welches etwa dem Aufwand eines PID-Regler entspricht, kann man etwa 25-30 msec Rechenzeit ansetzen, worin einige Summenpunkte, Wandleranschlüsse usw. bereits enthalten sind.

9. Zusammenfassung

In dieser Arbeit wurde die exemplarische Konstruktion eines umfangreichen Programmsystems für regelungstechnische Aufgaben dargestellt. Bereits in seiner grundlegenden Konzeption und in seiner Gliederung beruht dieses Programmsystem auf einer Situation, die als typisch für moderne Mikrorechner angesehen werden kann. Hierzu gehören vor allem Begrenzungen hinsichtlich der verfügbaren Speichermengen im Rechner selbst als auch auf den Massenspeichern. Auch wenn diese in Zukunft noch erheblich ansteigen, so werden vergleichbare Computer auch in der nächsten absehbaren Zukunft verbreitet sein - dann vielleicht im Taschenformat oder nur wenig größer, um auch graphische Darstellungsformen mit der nötigen Qualität sicherzustellen.

Die gesamte Arbeit zeigt, daß man einen erheblichen Grad an Unterstützung auch für diese kleinen Rechner erzielen kann. Moderne Softwarekonzepte mit interaktiver Kommunikation stellen in allen Fällen eine ganz wesentliche Forderung an solche Systeme dar. Es hat sich im Verlauf dieser Arbeit gezeigt, daß Mikrorechner hier nur unwesentlich benachteiligt sind gegenüber größeren Rechenanlagen.

Die vorausgesetzte Beschränkung auf weitverbreitete Standardbetriebssysteme ergab zwangsläufig Probleme bei der Realisierung paralleler Prozesse, wie sie für den "on-line"-Betrieb von Reglern zwingend erforderlich sind. Die Verwaltung der Echtzeit in den Betriebssystemen erscheint noch nicht ausreichend weit fortgeschritten. Es bleibt zu hoffen, daß hier in Zukunft Ergänzungen stattfinden, die die Echtzeitprogrammierung erleichtern. Zur Zeit wird man nicht umhinkommen, mindestens ein kleines hardwareabhängiges Programm vorzusehen, welches die erforderliche Schnittstelle definiert. Eine zweite wesentliche Eigenschaft, die bei Mikrorechnern zur Zeit noch uneinheitlich

unterstützt wird, stellt die graphische Darstellung dar. Nachdem es inzwischen zu einer GKS-Norm gekommen ist, und graphische Programme auch in nichttechnischen Anwendungen beachtlich zunehmen, kann man vermuten, daß dieser Mangel schon in sehr naher Zukunft nicht mehr gegeben sein wird.

Von den beiden angeführten Begrenzungen abgesehen, ist es weitgehend problemlos möglich, interaktive Software in leicht portabler Form zu erstellen. Eine Selbstbeschränkung auf Eigenschaften, die jeder Rechner heute besitzt, sollte hierzu als wichtigste Vorsorge angesehen werden.

Das gesamte Softwarepaket umfaßt zum augenblicklichen Zeitpunkt über 3 Megabyte ausführbare Objekte. Quellen und Dokumentationen besitzen jeweils etwa den gleichen Umfang. Hiermit wird bereits ein ansehnlicher Teil von Grundaufgaben der Regelungstechnik abgedeckt. Fast in allen Fällen bietet das System mehrere prinzipielle Möglichkeiten zur Wahl an. Innerhalb einer solchen sind häufig noch Optionen vorhanden, die eine zielgerichtete Anpassung oder Modifikation erlauben.

Da das System von Anbeginn als ein offenes System geplant war, steht es für eine Vielfalt von Ergänzungen zur Verfügung. Die bereits vorhandenen Teile können hierzu jederzeit als Module der bestehenden Software entnommen werden. Daß eine solche Vorgehensweise die Arbeit ganz erheblich erleichtert, ist unmittelbar einsichtig und hat sich im Laufe der Entwicklung immer wieder gezeigt.

Zur Analyse stehen statistische Verfahren zur Verfügung, die an mehreren Prozessen erfolgreich erprobt wurden. Die dabei zugrundeliegende Meßwert-erfassung wurde mehrfach eingesetzt und stellt eine einfache Methode dar, Information über das Verhalten eines Systems zu gewinnen.

Zum Entwurf einer Regelung sind jeweils mehrere Methoden für dynamische Regler, Zustandsregler und Beobachter vorhanden. Bei der Verwendung von Zustandsraummethoden konnte ein Teil des Entwurfs automatisiert werden. Hierzu wurden ausgehend von einer zugrundeliegenden Regelstreckenbeschreibung zwei modifizierte Zustandsraumdarstellungen ermittelt, die für den Beobachter- bzw. den Reglerentwurf verwendet werden. Anschließend werden beide Teilentwürfe wieder zusammengefaßt zu einer Zustandsregelung. Die entworfenen Regelungen, Teile hiervon oder auch nur die Regelstrecke allein, können mit verschiedenen Methoden simuliert werden. Anhand des zeitlichen Verhaltens der Regelstrecke als Reaktion auf definierte Eingriffe durch Stell- oder Störgrößen kann eine Beurteilung der Resultate erfolgen.

Zuletzt können die entworfenen Regler unmittelbar an der Regelstrecke eingesetzt werden. Hierfür sind mehrere Echtzeitregelprogramme vorhanden, die für die jeweilige Aufgabe konfiguriert werden können.

Mit gutem Erfolg konnte sowohl für direkte digitale Regelung als auch zum Zwecke der Simulation ein interaktives, blockorientiertes Konzept entwickelt und realisiert werden. Seiner Struktur nach ist es nicht auf lineare Systeme beschränkt, sondern enthält auch nichtlineare Blöcke. In Grenzen erlaubt es sogar binäre Operationen, obgleich es sehr spezialisierte Forderungen der Steuerungstechnik nicht berücksichtigt.

Als ein offenes System birgt das in dieser Arbeit beschriebene Softwarepaket zwangsläufig eine Forderung nach Ergänzung. Da es bereits an sehr verschiedenen Problemen auch von Dritten angewendet wurde, kann davon ausgegangen werden, daß hiermit eine zukunftssträchtige Entwicklung begonnen wurde.

Literaturverzeichnis

- A1 Ackermann, J.; Abtastregelung, Springer, Berlin, Heidelberg, New York, 1972
- A2 Ackermann, J.; Abtastregelung, Analyse und Synthese, Band I, Entwurf robuster Systeme, Band II, Springer, Berlin, Heidelberg, New York, 1983
- A3 Abel, B.; Algebraischer Reglerentwurf im Frequenzbereich, Diplomarbeit ESR 8307, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1983
- A4 Advanced Micro Devices Inc.; Am 9511 Datenblatt, 1977
- A5 Advanced Micro Devices Inc.; Am 9512 Datenblatt, 1979
- A6 Abel, B.; Prozeßdatenerfassung, Studienarbeit ESR 8213, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1982
- A7 Arndt, G.; Ein Beitrag zum Entwurf zeitdiskreter Regler für kombiniertes Stör- und Führungsverhalten linearer Tiefpaßregelstrecken, Dissertation, Technische Hochschule Aachen, Fakultät für Maschinenwesen, Aachen, 1979
- A8 Asselmeyer, B.; Eschwei, W.; Gräser, A.; Tolle, H.; A Concept for a CAD Program-Package for the Control-Engineer Using only small Computers, Computer Aided Design of Control Systems, Proceedings of the IFAC-Symposium, Zürich, Switzerland, 29-31 August 1979, Pergamon Press, 1980
- A9 Aström, K. J.; A SIMNON Tutorial, Department of Automatic Control, Lund Institute of Technology, Schweden, 1982

- B1 van den Bosch, P.P.J.; PSI - An Extended Interactive Block-oriented Simulation Program, Computer Aided Design of Control Systems, Proceedings of the IFAC-Symposium, Zürich, Switzerland, 29-31 August 1979, Pergamon Press, 1980
- B2 Bender, K.; Heinzel, W.; Jacob, H.; Motsch, W.; Weber, W.; Mikrorechner, VDI-Verlag, Düsseldorf, 1977
- B3 Biedermann, H.; Mikrocomputer und Publikation, UTB 1265, Fischer-Verlag, Stuttgart, 1984
- B4 Bittner, G.; Böhmer, B.; Lange, W.; Mäder, H. F.; Rosenau, L.; Timmes, R.; Neuartiges Antriebssystem für Großantennen, Tech. Mitteilungen Krupp, Forschungsberichte, Band 40 (1982), Heft 2, pp. 47 - 60
- B5 Böttiger, F.; Untersuchung von Kompensationsalgorithmen für die direkte digitale Regelung, KFK-PDV Bericht 146, Juni 1978, Gesellschaft für Kernforschung mbH, Karlsruhe
- B6 Böttiger, F.; Untersuchung von DDC-Algorithmen mit vorgegebener Struktur, KFK-PDV Bericht 147, Juni 1978, Gesellschaft für Kernforschung mbH, Karlsruhe
- B7 Buxbaum, A.; Schierau, K.; Berechnung von Regelkreisen in der Antriebstechnik, AEG Telefunken, Berlin, 1980
- B8 Bergen, A. R.; Ragazzini, J. R.; Sampled Data Processing Techniques for Feedback Control Systems, AIEE Transactions, Part II, 73, 1954, pp. 236 - 247

- B9 Bongiorno, J. J.; Youla, D. C.; On Observers in Multivariable Control Systems, Int. Journ. Control, Vol. 8, Nr. 3, 1968, pp. 221 - 243
- B10 Bellman, R.; Introduction to Matrix Analysis, McGraw Hill, 1960
- B11 Brigham, E. O.; The Fast Fourier Transform, Englewood Cliffs, N.J., Prentice-Hall, 1974
- B12 van den Bosch, P.P.J.; PSI, Benutzermanual, Delft University of Technology, Department of Electrical Engineering, Delft, 1981
- C1 Cellier, F. E.; Combined Continuous/Diskrete System Simulation by Use of Digital Computers: Techniques and Tools; Dissertation ETH 6483, Swiss Federal Institute of Technology, Zürich, 1979
- C2 Cromemco Inc.; Cromix Operating System Manual, 1980
- C3 Cohen, G. H.; Coon, G. A.; Theoretical Considerations of Retarded Control, Taylor Instr. Comp. Bulletin, TOS-10A102
- C4 Cumming, S. D. G.; Design of Observers of Reduced Dynamics, Electronics Letters, Vol. 5, 1969, No. 10, pp. 213 - 214
- C5 Cooley, J. W.; Tukey, J. W.; An Algorithm for the Machine Calculation of Complex Fourier Series, Math. Computation, Vol. 19, 1965
- C6 Cuenod, M. A.; Computer Aided Design of Control Systems, Proceedings of the IFAC-Symposium, Zürich, Switzerland, 29-31 August 1979, Pergamon Press, 1980

- D1 Digital Research; MP/M Multi-Programming Monitor Control Program, User's Guide, 2nd Printing, 1980

- D2 Dongarra, J.; Moler, C.; Bunch, J.; Stewart, G.; LINPACK User's Guide, SIAM, 1979

- D3 Diekmann, K.; Unbehauen, H.; On-line parameter estimation in a class of nonlinear systems via modified least-squares and instrumental variable algorithms, 7th IFAC-Symposium on Identification and System Parameter Estimation, 3-7 July 1985, University of York, United Kingdom

- E1 Enderle, G.; Kansy, K.; Pfaff, G.; Computer Graphics Programming, Springer, Berlin, 1984

- F1 Färber, G.; Prozeßrechentchnik, Springer, Berlin, Heidelberg, 1979

- F2 Forner, M.; CSMP - Blockorientierte Sprachen zur digitalen Simulation dynamischer Systeme, IBM - Nachrichten 18 (1968), Nr. 187, pp. 51 - 57

- F3 Färber, G.; Status of Hardware and Software for Mircocomputers, Preprints 6th IFAC/IFIP Conference, Digital Computer Applications to Process Control, Düsseldorf, 14-17 Okt. 1980, pp. 1 - 10

- F4 Föllinger, O.; Lineare Abtastsysteme, Oldenbourg, München, 1974

- F5 Föllinger, O.; Entwurf von Abtastregelungen, ETZ-A, 1967, Heft 6, pp. 150 - 155

- F6 Föllinger, O.; Regelungstechnik, AEG Telefunken, 3. Auflage, Berlin, 1980

- F7 Föllinger, O.; Synthese von Abtastsystemen im Zeitbereich, Regelungstechnik und Prozeß-Datenverarbeitung, Jahrg. 13, 1965, Nr. 6, pp. 269 - 275

- G1 Garbow, B. S.; Boyle, J. M.; Dongarra, J. J.; Moler, C. B.; Matrix Eigensystem Routines - EISPACK Guide Extension, Springer, Berlin, Heidelberg, New York, 1977
- G2 Giloi, W. R.; Principles of Continuous System Simulation, Teubner, Stuttgart, 1975
- G3 Guzaitis, J.; Concurrent CP/M, Digital Research Inc, 1983 und BYTE magazine, Nov. 1983
- G4 Gregory, R. T.; Karney, D. L.; A Collection of Matrices for Testing Computational Algorithms, Wiley-Interscience, New York, 1969
- G5 Gopinath, B.; On the Control of Linear Multiple Input-Output Systems, Bell System Techn. Journal, Vol. 50, Nr. 3, pp. 1063 - 1081, 1971
- G6 Gulbins, J.; UNIX, Springer, Berlin, 1984
- G7 Grübel, G.; Joos, H. D.; Höfler, A. B.; Regelungstechnische Analyse- und Synthese-Programme RASP, Aussprachetag 26./27. Sept. 1983, Langen, Rechnergestützter Regelkreisentwurf, VDI/VDE-Gesellschaft Meß- und Regelungstechnik, Düsseldorf, 1983
- H1 Haase, F.; MP/M Systeminterface für Pascal Echtzeitprogramme, Interner Bericht ESR 8304, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1983
- H2 Haase, F.; Analoges und digitales Prozeßinterface, Interner Bericht ESR 8302, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1983

- H3 Haase, F.; Geppert, H.; Blockorientierte Simulation dynamischer Systeme, Interner Bericht ESR 8404, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1984
- H4 Haase, F.; Interactive Blockorientated Control by Microcomputer, IFAC Symposium on Theory and Application of Digital Control, New Delhi, India, Jan. 5-7, 1982
- H5 Haase, F.; RTE 3 - Echtzeitbetriebssoftware für einen Mikrorechner, Interner Bericht ESR 8108, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1981
- H6 Haase, F.; Entwurf deterministischer Beobachter, Studienarbeit ESR 7704, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1977
- H7 Hain, K.; Digitale Simulation von Regelkreisen mittels Übertragungsfunktionen, Diplomarbeit ESR 8317, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1984
- H8 Haase, F.; Abel, B.; Basis-Interface zum Lesen und Schreiben von Standarddateien, Interner Bericht ESR 8103, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1983
- H9 Haase, F.; Matrizen-Manager, Interner Bericht ESR 8103, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1983
- I1 Isermann, R.; Digital Control Systems, Springer, Berlin, 1981
- I2 Intel Corporation; OEM Systems Handbook, 1983
- I3 Intel Corporation; The 8086 User's Manual, 1979

- I4 Intel Corporation; iAPX 286/10 High Performance Microprocessor with Memory Management and Protection, Advance Information, 1982
- I5 Intel Corporation; iAPX 86/20 Numeric Data Processor, Preliminary Datasheet, 1981
- I6 Institute of Measurement and Control, Workshop Computer Aided Control System Design, University of Sussex, Brighton, UK, 19-21 September, 1984
- I7 Isermann, R.; et al; Comparison and evaluation of six on-line identification and parameter estimation methods with three simulated processes, Proc. IFAC-Symposium, Hague, 1973
- I8 Isermann, R.; Rechnerunterstützter Entwurf digitaler Regelungen auf der Grundlage identifizierter Prozessmodelle, Aussprachetag 26./27. Sept. 1983, Langen, Rechnergestützter Regelkreisentwurf, VDI/VDE-Gesellschaft Meß- und Regelungstechnik, Düsseldorf, 1983
- J1 Jensen, K.; Wirth, N.; Pascal User Manual and Report, Springer, New York, Heidelberg, Berlin, 1978
- J2 Johnson, C. D.; Accomodations of Disturbances in Optimal Control Problems, Int. Journ. Control, Vol. 15, 1972, pp. 209 - 231
- J3 Johnson, C. D.; On Observers for Systems with Unknown and Inaccessible Inputs, Int. Journ. Control, Vol. 21, 1975, pp. 825 - 831
- J4 Jürgens, K.; Mikrorechner-Regelung eines fremd-erregten Gleichstrommotors, Diplomarbeit ESR 8403, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1984

- J5 Jamshidi, M.; Cuneo, P.; TIMDOM/PC User"s Guide - Software for Multivariable Control Systems, Report No. LCAD-84-01, Laboratory for Computer-Aided Design of Systems and Networks, Department of Electrical and Computer Engineering, The University of New Mexico, Albuquerque, USA, 1984

- K1 Kreisselmeier, G.; A Solution of the Bilinear Matrix Equation $\underline{A}\underline{Y} + \underline{Y}\underline{B} = -\underline{Q}$, SIAM Journ. Appl. Math., Vol. 23, 1972, pp. 334 - 338

- K2 Köckemann, A.; Zustandsregelung mit einem Mikrorechner, Diplomarbeit ESR 8306, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1983

- K3 Kortmann, M.; Die Identifikation technischer Anlagen mit Hilfe eines Mikrorechners, Diplomarbeit ESR 8401, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1983

- K4 Komolka, J.; Beobachterentwurf auf einem Mikrorechner, Diplomarbeit ESR 8313, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1984

- K5 Korn, G. A.; Wait, J. V.; Digital Continuous-System Simulation, Prentice-Hall, Englewood-Cliffs, New Jersey, 1978

- L1 Leonhard, W.; Zur Anwendung von Digitalrechnern als Abtastregler, Archiv Elektrotechnik 51, 1966, Nr. 14, pp 72 - 75

- L2 Ludyk, G.; Theorie dynamischer Systeme, Elitera, Berlin, 1977

- L3 Laub, A. J.; Computational Methods in Control - A Survey and Introduction to the Literature, 8th IFAC Congress Preprints, Control Science and Technology, Aug. 24-28, Kyoto, Japan, 1981
- L4 Luenberger, D. G.; Observing the State of a Linear System, IEEE Trans. Milit. Electr. Mil-8, Nr. 2, 1964, pp. 74 - 80
- L5 Luenberger, D. G.; Observers for Multivariable Systems, IEEE Trans. Autom. Control, AC-11, Nr. 2, 1966, pp. 190 - 197
- L6 Leonhard, W.; Statistische Analyse linearer Regelsysteme, Teubner Studienbücher, Stuttgart, 1973
- M1 MT MicroSystems, Pascal MT+ Manual, New York, 1980
- M2 Motorola Inc.; M68000 Real-Time Multitasking Software User's Guide, 1980
- M3 Magyar, P.; Schnieder, E.; Vollstedt, W.; Digitale Regelung und Steuerung einer stromrichter-gespeisten Gleichstrommaschine mit Mikrorechner, Regelungstechnik, 30. Jahrg., 1982, pp. 378 - 387
- M4 Martens, H. R.; Jaliwalla, S. A.; The Design of digital Algorithms for Process Control, JACC 1973, Columbus, Ohio, paper 13.5, pp. 375 - 384
- M5 Moore, C. F.; Smith, C. L.; Murill, P. W.; Improved Algorithms for Direct Digital Control, Instr. and Control Systems 43, 1970, Nr. 1, pp. 70 - 74
- M6 Meditch, J. S., Hostetter, G. H.; Observers for Systems with Unknown and Inaccessible Inputs, Int. Journ. Control, Vol. 19, 1974, pp. 473 - 480

- M7 Morel, R. S.; FREEDOM-TIMDOM/45: A Descriptive Guide, Report No. LCAD-84-02, Department of Electrical and Computer Engineering, The University of New Mexico, Albuquerque, USA, 1984

- N1 Naslin, P.; Dynamik linearer und nichtlinearer Systeme, Oldenbourg, München, 1968

- N2 Neudecker, H.; A Note on Kronecker Matrix Products and Matrix Equation Systems, SIAM Journ. Appl. Math., Vol. 17, 1969, No. 3

- P1 Preuß, H. P.; Entwurf stationär genauer Zustandsrückführungen, Regelungstechnik, 28. Jahrg., H. 2, 1980, pp. 51 - 56

- R1 Rimval, M.; On the Use of Ada in CACSD, Workshop Computer Aided Control System Design, Institute of Measurement and Control, University of Sussex, Brighton, UK, 19-21 September, 1984

- R2 Rosenbrock, H. H.; State-Space and Multivariable Theory, T. Nelson and Sons Ltd., London, 1970

- S1 Smith, B. T.; Boyle, J. M.; Dongarra, J. J.; Garbow, B. S.; Ikebe, Y.; Klema, V. C.; Moler, C. B.; Matrix Eigensystem Routines - EISPACK Guide, 2nd Edition, Springer, Berlin, Heidelberg, New York, 1976

- S2 Schmid, Chr.; Interaktive Benutzung des KEDDC-Systems CSMP-1000, Interner Bericht ESR 8003, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1980

- S3 Strejc, V.; Synthese von Regelkreisen mit Prozeßrechnern, Akademie-Verlag, Berlin, 1967

- S4 Sondermann, H.; Erweiterung des Programmsystems "Korrelationsanalyse" zur Bestimmung des Frequenzganges auf dem Mikrorechner MC6800, Diplomarbeit ESR 7917, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1980

- S5 Schmid, C.; Juen, G.; Design of Digital State-Feedback Controllers Using a CAD Package, IFAC Symposium on Theory and Application of Digital Control, New Dehli, India, January 5-7, 1982

- S6 Sczegan, G.; Entwicklung eines Programmsystems "Korrelationsanalyse" für den Mikrorechner MC6800, Diplomarbeit ESR 7808, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1979

- S7 Schmidt, G.; MSRBASIC, Kurzübersicht, Lehrstuhl für Steuerungs- und Regelungstechnik, Technische Universität München, 1984

- T1 Thomas, R.; Yates, J.; A User Guide to the UNIX System, Osborne/McGraw-Hill, Berkeley, California, 1982

- T2 Tröndle, H. P.; Anwenderorientierte Auslegung von Abtastreglern nach der Methode der Doppelverhältnisse, Regelungstechnik, 12. Jahrg., 1978, pp. 384 - 391

- T3 Takahashi, Y.; Chan, C. S.; Auslander, D. M.; Parametereinstellung bei linearen DDC-Algorithmen, Regelungstechnik, 19. Jahrg., 1971, Nr. 6, pp. 237 - 244

- U1 Unbehauen, H.; et al; Simulationstechnik, Regelungstechnisches Seminar WS 81/82, Interner Bericht ESR 8215, Lehrstuhl für - elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1982

- U2 Unbehauen, H.; et al; Einsatz von Mikrorechnern in der Regelungs- und Steuerungstechnik, Regelungstechnisches Seminar SS 84, Interner Bericht ESR 8414, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1984

- U3 Unbehauen, H.; Regelungstechnik I, Vieweg, Braunschweig, Wiesbaden, 1982

- U4 Unbehauen, H.; Regelungstechnik II, Vieweg, Braunschweig, Wiesbaden, 1983

- U5 Unbehauen, H.; Regelungstechnik III, Vieweg, Braunschweig, Wiesbaden, 1985

- U6 Unbehauen, H.; et al; Digitale Regelsysteme, Regelungstechnisches Seminar SS 83, Interner Bericht ESR 8317, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1983

- W1 Wiener, R.; Sincovec, R.; Programming in Ada, John Wiley & Sons, 1983

- W2 Werum, R.; Windauer, H.; Pearl, Vieweg, 1978

- W3 Wortmann, W.; Zustandsreglerentwurf auf einem Mikrorechner, Diplomarbeit ESR 8312, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1984

- W4 Wortmann, W.; Komolka, J.; DDC - Interaktive digitale Regelung in blockorientierter Darstellung, Interner Bericht ESR 8405, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1984

- W5 Wilkinson, J. H.; Reinsch, C.; Lineare Algebra, Vol. II, Springer, Berlin, 1971

- W6 Weber, E.; Erprobung verschiedener Regelungsverfahren zur Regelung eines stehenden Pendels, Studienarbeit ESR 7913, Lehrstuhl für elektrische Steuerung und Regelung, Ruhr Universität Bochum, 1980
- Y1 Young, P.; Shellswell, S.; Neethling, C.; A recursive approach to time-series analysis, Report cued/b-Control/TR 16, University Cambridge, 1971
- Z1 Zaks, R.; Einführung in Pascal und UCSD-Pascal, Sybex, Düsseldorf, 1982
- Z2 Ziegler, J. G.; Nichols, N. B.; Optimum Settings for Automatic Controllers, Trans. ASME 64, 1942, pp. 759 -765
- Z3 Zurmühl, R.; Matrizen, Springer, Berlin, 1964
- Z4 Zimmermann, R.; Dead-beat-Regleralgorithmen für Mikroprozessor-Regler, Messen - Steuern - Regeln, 23. Jahrg., 1980, H.2, pp. 68 - 72

Zusammenstellung der wichtigsten Formelzeichen

<u>A</u>	Systemmatrix
A(z)	Nennerpolynom
<u>B</u>	Eingangs- oder Steuermatrix
<u>B_z</u>	Störeingangsmatrix, meßbar
<u>B_{zn}</u>	Störeingangsmatrix, nicht meßbar
B(z)	Zählerpolynom
<u>B⁺</u>	Moore-Penrose-Inverse zu <u>B</u>
<u>C</u>	Ausgangs- oder Beobachtungsmatrix
<u>D</u>	Systemmatrix des Beobachters
<u>E</u>	Eingangsmatrix des Beobachters für den Regelstreckenausgang
e	Fehlersignal der "Least-Squares"-Methode Regelabweichung
<u>F</u>	Rückführmatrix
<u>G</u>	Eingangsmatrix des Beobachters für den Regelstreckeneingang
G(s/z)	Übertragungsfunktion
K _w	Führungsverhalten
<u>M</u>	Bewertungsmatrix
<u>m</u>	Meßvektor
<u>P</u>	Kovarianzmatrix
P(z)	dynamisches Vorfilter
<u>p</u>	Parametervektor
<u>Q</u>	Bewertungsmatrix
Q(z)	Filter im Rückkopplungspfad
<u>R</u>	Bewertungsmatrix
R _{xx}	Autokorrelationsfunktion
R(z)	Reglerübertragungsfunktion
R _{xy}	Kreuzkorrelationsfunktion
<u>T</u>	Beobachtertransformationsmatrix
<u>u</u>	Eingangsvektor
<u>v</u>	Beobachterzustandsvektor Fehlersignal der Hilfsvariablenmethode
<u>x</u>	Zustandsvektor
<u>y</u>	Ausgangsvektor

Lebenslauf

21. Mai 1948 geboren in Duisburg-Rheinhausen als Sohn des Karl Haase und seiner Frau Johanna Maria, geb. Flick
- 1954 - 1958 Besuch von Grundschulen in Rheinhausen
- 1958 - 1968 Besuch der Privatschule Zimmermann in Duisburg
- 1968 Abitur vor der Prüfungskommission des Regierungspräsidenten in Düsseldorf
- 1968 - 1969 Bundeswehr
- 1969 - 1972 Studium der Mathematik in Tübingen und Bochum
- 1972 - 1979 Studium der Elektrotechnik in Bochum
- Januar 1972 Heirat mit Regina Haase, geb. Letsch
- August 1973 Geburt des ersten Kindes Cor Haase
- Januar 1975 Geburt des zweiten Kindes Glyn Haase
- Mai 1979 Diplom Elektrotechnik
- seit 1979 wissenschaftlicher Assistent am Lehrstuhl für elektrische Steuerung und Regelung an der Ruhr Universität Bochum
- Bochum, im April 1985

